INTERNATIONAL STANDARD

ISO/IEC 9506-1

First edition
1990-10-15

Industrial automation systems — Manufacturing Message Specification —

Part 1: Service definition

Systèmes d'automatisation industrielle — Spécification de messagerie industrielle — Partie 1 : Définition de service

Reference number
ISO 9506-1 : 1990 (E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO/IEC 9506-1 was prepared by Technical Committee ISO/IEC TC 184, *Industrial automation systems*, Sub-Committee SC 5, *System integration and communication.*

ISO/IEC 9506 consists of the following parts, under the general title *Industrial automation systems — Manufacturing Message Specification*:

— *Part 1: Service definition*

— *Part 2: Protocol specification*

Introduction

This part of ISO/IEC 9506 provides a wide variety of services useful for various manufacturing and process control devices. It is designed to be used both by itself and in conjunction with Companion Standards, which describe the application of subsets of these services to particular device types.

The services provided by the Manufacturing Message Specification (MMS) range from simple to highly complex. It is not expected that all of these services will be supported by all devices. The subset to be supported is limited in some cases by Companion Standards, and in all cases may be limited by the implementor. Characteristics important in selection of a subset of services to be supported include:

a) applicability of the service to the device;

b) the complexity of services and requirements;

c) the complexity of provision of a particular class of service via the network versus the complexity of the device.

Security considerations

When implementing MMS in secure or safety critical applications, features of the OSI security architecture may need to be implemented. Appropriate features should be selected from ISO 7498-2 covering safety architectures and features. Those of particular interest cover the position (in OSI) of

a) access control;

b) authentication;

c) non-repudiation.

Specific implementation methods shall be at the discretion of the implementor.

Complexity of services and requirements

Some MMS services are quite complex and should be considered as advanced functions. Devices used in very simple applications often will not require such advanced functions, and hence will not support such MMS services.

Keywords

Application Interworking
Application Layer Protocol
Information Processing Systems
Manufacturing Communications Network
Manufacturing Message Specification
Numerical Control System
Open Systems Interconnection
OSI Reference Model
Process Control System
Programmable Controller
Programmable Device
Robotics Control System
Virtual Manufacturing Device

General

This part of ISO/IEC 9506 is one of a set of International Standards developed to facilitate the interconnection of information processing systems. It is positioned within the application layer of the Open Systems Interconnection Environment as an Application Service Element (ASE) with respect to other related standards by the Basic Reference Model for Open Systems Interconnection (ISO 7498).

The aim of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of information processing systems:

a) from different manufacturers;

b) under different managements;

c) of different levels of complexity;

d) of different ages.

Purpose

The purpose of this part of ISO/IEC 9506 is to define the service provided by the Manufacturing Message Specification. The MMS Service is provided by the Manufacturing Message Specification Protocol making use of services available from the Association Control Service Element (ASCE) and the Presentation layer, as defined in ISO 8649 and ISO 8822, respectively.

This part of ISO/IEC 9506 is concerned, in particular, with the communication and interworking of programmable manufacturing devices. By using this standard together with other standards positioned within the OSI Reference Model, otherwise incompatible systems may work together in any combination.

ISO/IEC 9506-2 specifies the protocol that supports the Manufacturing Message Specification.

This page intentionally left blank

Industrial automation systems - Manufacturing Message Specification - Part 1: Service definition

1 Scope

The Manufacturing Message Specification is an application layer standard designed to support messaging communications to and from programmable devices in a Computer Integrated Manufacturing (CIM) environment. This environment is referred to in ISO/IEC 9506 as the manufacturing environment. This part of ISO/IEC 9506 does not specify a complete set of services for remote programming of devices, although provision of such a set of services may be the subject of future standardization efforts.

This part of ISO/IEC 9506 defines the Manufacturing Message Specification within the OSI application layer in terms of

a) an abstract model defining the interaction between users of the service;

b) the externally visible functionality of implementations conforming to ISO/IEC 9506, in the form of procedural requirements associated with the execution of service requests;

c) the primitive actions and events of the service;

d) the parameter data associated with each primitive action and event;

e) the relationship between, and the valid sequences of, these actions and events.

The service defined in this part of ISO/IEC 9506 is that which is provided by the Manufacturing Message Specification protocol. The service may be used by other application layer service elements or by other elements of the application process.

This part of ISO/IEC 9506 does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. This part of ISO/IEC 9506 specifies the externally visible functionality of implementations together with conformance requirements for such functionality.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9506. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9506 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 646 : 1983, *Information processing — ISO 7-bit coded character set for information interchange.*

ISO 7498 : 1984, *Information processing systems — Open Systems Interconnection — Basic Reference Model.*

ISO 7498-2 : 1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.*

ISO 7498-3 : 1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 3: Naming and addressing.*

ISO 8326 : 1987, *Information processing systems — Open Systems Interconnection — Basic connection oriented session service definition.*

ISO/TR 8509 : 1987, *Information processing systems — Open Systems Interconnection — Service conventions.*

ISO 8571 : 1988, *Information processing systems — Open Systems Interconnection — File Transfer, Access and Management.*

ISO 8649 : 1988, *Information processing systems — Open Systems Interconnection — Service definition for the Association Control Service Element.*

ISO 8650 : 1988, *Information processing systems — Open Systems Interconnection — Protocol specification for the Association Control Service Element.*

ISO 8822 : 1988, *Information processing systems — Open Systems Interconnection — Connection oriented presentation service definition.*

ISO 8824 : 1987, *Information processing systems — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).*

ISO 8824/Add 1 : — 1), *Information processing systems — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1) Addendum 1: ASN.1 Extensions.*

ISO 8825 : 1987, *Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 8825/Add 1 : — 1), *Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) Addendum 1: ASN.1 Extensions.*

ISO 9040 : — 1), *Information processing systems — Open Systems Interconnection — Virtual terminal service — Basic class.*

ISO 9041 : — 1), *Information processing systems — Open Systems Interconnection — Virtual terminal protocol — Basic class.*

ISO/IEC 9506-2 : 1990, *Industrial automation systems — Manufacturing Message Specification — Part 2: Protocol specification.*

ISO/IEC 9545-1 : 1989, *Information technology — Open Systems Interconnection — Application Layer Structure.*

ISO/IEC 9594 : — 1), *Information processing systems — Open Systems Interconnection — The Directory.*

IEEE 754 : 1985, *IEEE Standard for Binary Floating-Point Arithmetic.*

3 Definitions

NOTE – The definitions contained in this clause make use of abbreviations defined in clause 4.

For the purposes of this part of ISO/IEC 9506, the following definitions apply.

3.1 Reference Model definitions

This part of ISO/IEC 9506 is based on the concepts developed in the Basic Reference Model for Open Systems Interconnection (ISO 7498), and makes use of the following terms defined in that International standard:

a) application-entity;

b) application-process;

c) application service element;

d) open system;

e) (N)-protocol;

f) (N)-protocol-data-unit;

g) (N)-service-access-point;

h) (N)-layer;

i) system;

1) To be published.

j) (N)-user-data.

3.2 Service Convention definitions

This part of ISO/IEC 9506 makes use of the following terms defined in the OSI Service Conventions (ISO/TR 8509) as they apply to the Manufacturing Message Specification:

a) confirm;

b) indication;

c) primitive;

d) request;

e) response;

f) service primitive;

g) service provider;

h) service user.

3.3 Abstract Syntax Notation definitions

This part of ISO/IEC 9506 makes use of the following terms defined in the Abstract Syntax Notation One (ASN.1) Specification (ISO 8824):

1) value;

2) type;

3) simple type;

4) structure type;

5) component type;

6) tag;

7) tagging;

8) type (or value) reference name;

9) character string type;

10) boolean type;

11) true;

12) false;

13) integer type;

14) bitstring type;

15) octetstring type;

16) null type;

17) sequence type;

18) sequence-of type;

19) tagged type;

20) choice type;

21) selection type;

22) real type;

23) object identifier type;

24) module;

25) production;

26) ASN.1 encoding rules;

27) ASN.1 character set;

28) external type.

3.4 Other definitions

This part of ISO/IEC 9506 makes use of the following terms:

3.4.1 AA-specific (Application Association specific):

An adjective used to describe an object whose name has a scope that is a single application association (i.e. the name may be referenced only on the application association with respect to which the object was defined).

3.4.2 attribute:

A data element, having a defined meaning, together with a statement of the set of possible values it may take.

3.4.3 conformance building block (CBB):

An atomic unit used to describe MMS conformance requirements.

3.4.4 Called MMS-user:

The MMS-user that issues the Initiate.response service primitive.

3.4.5 Calling MMS-user:

The MMS-user that issues the Initiate.request service primitive.

3.4.6 Client:

The peer communicating entity which makes use of the VMD for some particular purpose via a service request instance.

3.4.7 data:

Any representation to which meaning is or might be assigned (e.g. characters).

3.4.8 domain:

An abstract object that represents a subset of the capabilities of a VMD which is used for a specific purpose.

3.4.9 Domain-specific:

An adjective used to describe an object whose name has a scope that is a single domain (i.e. the name can be referenced over all application associations established with the VMD that may reference this domain).

3.4.10 download:

The process of transferring the content of a domain, including any subordinate objects, via load data to an MMS-user.

3.4.11 event management:

The management of event conditions, event actions, and event enrollments.

3.4.12 file:

An unambiguously named collection of information having a common set of attributes.

3.4.13 file operation:

The transfer of files between open systems, the inspection, modification or replacement of part of a file's content, or the management of a file and its attributes.

3.4.14 filestore:

An organized collection of files, including their attributes and names, residing at a particular open system.

3.4.15 information:

The combination of data and the meaning that it conveys.

3.4.16 journal:

A set of recorded, time-tagged event transitions, variable data, and/or comments, which may be logically ordered during retrieval.

3.4.17 local matter:

A decision made by a system concerning its behaviour in the Manufacturing Message Specification that is not subject to the requirements of ISO/IEC 9506.

3.4.18 Manufacturing Message Protocol Machine (MMPM):

An abstract machine that carries out the procedures specified in this part of ISO/IEC 9506.

3.4.19 MMS-environment:

A specification of the service elements of MMS and semantics of communication to be used during the lifetime of an application association.

3.4.20 MMS-provider:

That part of the application entity that conceptually provides the MMS service through the exchange of MMS PDUs.

3.4.21 MMS-user:

That portion of the application process which conceptually invokes the Manufacturing Message Specification.

3.4.22 monitored event:

A detected change in the state of an event condition.

3.4.23 network-triggered event:

An event which occurs due to an explicit solicitation by a client.

3.4.24 operator station:

An abstract object representing equipment associated with a VMD that provides for input/output interaction with an operator.

3.4.25 predefined object:

An object, whose name is of VMD-specific, Domain-specific or Application Association-specific scope, that is instantiated through the use of some mechanism other than an MMS service.

3.4.26 program invocation:

An abstract object representing a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment, which is composed of a set of domains.

3.4.27 Receiving MMPM:

The MMPM that receives an MMS PDU.

3.4.28 Receiving MMS-user:

The MMS-user that receives an indication or confirmation service primitive.

3.4.29 remote device control and monitoring:

The manipulation or inspection of the state of a device attached to the responder of a service request.

3.4.30 semaphore:

A conceptual lock associated with a logical or physical resource that permits access to that resource only by an owner of the lock.

3.4.31 semaphore management:

The control of semaphores.

3.4.32 Server:

The peer communicating entity which behaves as a VMD for a particular service request instance.

3.4.33 Sending MMPM:

The MMPM that sends an MMS PDU.

3.4.34 Sending MMS-user:

The MMS-user issues a request or response service primitive.

3.4.35 standardized object:

An object instantiation, whose name is of VMD-specific or Domain-specific scope, whose definition is provided in this part of ISO/IEC 9506 or an MMS Companion Standard.

3.4.36 type:

An abstract description of a set of values which may be conveyed by the value of a variable.

3.4.37 upload:

The process of transferring the content of a domain, including any subordinate objects, via load data from a remote user, in such a manner as to allow subsequent download.

3.4.38 variable:

One or more data elements that are referred to together by a single name or description.

3.4.39 variable access:

The inspection or modification of variables or components of variables defined at a VMD.

3.4.40 VMD-specific:

An adjective used to describe an object whose name has a scope that is a single VMD (i.e. the name may be referenced by all application associations established with the VMD).

4 Abbreviations

AA	: application association
ACSE	: Association Control Service Element
AE	: application entity
AP	: application process
ASE	: application service element
ASN.1	: Abstract Syntax Notation One
CBB	: conformance building block
FRSM	: file read state machine
FTAM	: File Transfer, Access and Management
MMPM	: Manufacturing Message Protocol Machine
MMS	: Manufacturing Message Specification
NC	: Numerical Control
OSI	: Open Systems Interconnection
PC	: Programmable Controller
PDU	: protocol data unit
PSAP	: presentation service access point
SAP	: service access point
SDU	: service data unit
ULSM	: upload state machine
VMD	: Virtual Manufacturing Device
VT	: Virtual Terminal

5 Conventions

5.1 Base of Numeric Values

This part of ISO/IEC 9506 uses a decimal representation for all numeric values unless otherwise noted.

5.2 Service Parameter Description

This part of ISO/IEC 9506 uses a tabular format to describe the component parameters of the MMS service primitives. Each table consists of up to six columns, containing the name of the service parameter, a column each for the request ("Req"), indication ("Ind"), response ("Rsp"), and confirm ("Cnf") primitives, and a column for conformance building block specification ("CBB"). The "Rsp" and "Cnf" columns are absent when the service is not a confirmed service.

5.2.1 Companion Standard Service Parameters

Some MMS service procedures which allow extensions to be supplied by Companion Standards. Such service procedures allow the specification of extra parameters in either the request primitive or in the response (+) primitive or both. This is indicated in the tables by the appearance of "COMP" following the "Argument" or the "Result(+)" entry of the service table respectively. Some service procedures allow the specification of additional parameters by the Companion Standards directly in an Argument list. Such occurrences will be indicated by the "COMP" in the columns describing the service primitives.

5.2.2 Service Table Structure

For those tables that require support of particular parameter conformance building blocks, the required conformance building blocks are enumerated on the first line in the table. In the remainder of each table, one parameter (or part of it) is listed on each horizontal line. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the vertical column:

M - parameter is mandatory for the primitive

U - parameter is a user option, and may or may not be provided depending on dynamic usage by the MMS-user

C - parameter is conditional upon other parameters or the environment of the MMS-user

- (blank) parameter is never present

COMP - parameter is for definition in MMS Companion Standards. Such parameters shall not be used other than as defined in a Companion Standard (and shall be omitted in the abstract syntax defined in Clause 19 of 9506-2). Annex C provides additional detail on the use of Companion Standards.

S - parameter is a selection from a collection of two or more possible parameters. The parameters that make up this collection are indicated in the table as follows:

a) each parameter in the collection is specified with the code "S";

b) the name of each parameter in the collection is at the same indentation from the beginning of the parameter column in the table;

c) Either
   1) each parameter is at the leftmost (outer) indentation in the table; or
   2) each parameter is part of the same parameter group. A parameter group is a collection of parameters where each group member has a common parent parameter. The parent parameter for any group member is the first parameter above the member that is not indented as far as that member. In the example below, ParameterA and ParameterB form a parameter group:

```
ParameterX
   ParameterA
   ParameterB
ParameterY
   ParameterC
```

Informally, for parameters involved in a selection, the indentation in the services tables signifies which parameters are involved in a selection. All parameters at the same level of indentation that are under a common "higher level" parameter are a part of the same selection.

The code "(=)" following one of the codes M, U, C or S indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. (For instance, an "M(=)" code in the indication service primitive column and an "M" in the request service primitive column means that the parameter in the indication primitive is semantically equivalent to that in the request primitive.)

Some parameters may contain subparameters. Subparameters are indicated by labelling of the parameter as M, U or C, and indenting all subparameters under the parameter. Presence of subparameters is always dependent on presence of the parameter that they appear under (for example, an optional parameter may have subparameters; if the parameter is not supplied, then no subparameters may be supplied).

The CBB column is used to indicate that usage of the parameter is dependent on support of conformance building blocks, other than that containing the service. If no entry exists in the CBB column, then there is no dependency on other conformance building blocks. If an entry does exist, then the parameter is available (and permitted for use by this Standard) if and only if the named conformance building block is supported and negotiated for use.

Some service parameters are named using a "List Of . . . " convention. Unless otherwise noted, all parameters whose names begin with "List Of . . . " specify a list of zero or more of the item specified after the "List Of" keyword phrase. (This type of parameter corresponds with the sequence-of ASN.1 type in ISO/IEC 9506-2.)

The descriptions of parameters in this part of ISO/IEC 9506 make reference to types, in order to describe the allowable values for such parameters. The types referenced may either be types defined in ISO 8824 (Abstract Syntax Notation One), or may be defined in ISO/IEC 9506-2.

5.3 Invocation Identifier on Service Primitives

For services identified in the ConfirmedServiceRequest production in 7.5.2 of ISO/IEC 9506-2, each MMS service primitive contains an "Invoke ID" parameter, which is mandatory in the request, indication, response, and confirm primitives. The value in the indication, response, and confirm primitives is semantically equivalent to that in the request primitive. This parameter serves to identify unambiguously the service invocation from an MMS-user on an application association. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

5.4 List Of Modifier on Service Primitives

Every confirmed MMS service contains a "List Of Modifier" parameter, which is a user option in the request and indication primitives. The value in the indication primitive is semantically equivalent to that in the request primitive. This parameter serves to specify a list of one or more service state machine modifiers which add a condition which must be satisfied for the execution of the service request to begin. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

MMS defines two modifiers: the AttachToSemaphore modifier and the AttachToEventCondition modifier, which are described in clauses 13 and 15, respectively.

The effect of the modifier on the state machine for execution of a confirmed MMS service is described in clause 7.

5.5 Addressing in MMS

The MMS Standard does not provide the means for naming and addressing of a peer MMS-user or peer Manufacturing Message Protocol Machine (MMPM). Within OSI, the identification and addressing of peer application entities is carried out through the use of ACSE services defined in ISO 8649. After such association of peers has succeeded, all of the MMS PDUs flow between these peers over the established presentation connection. It is therefore not necessary for MMS to carry addressing information. Additional information on naming and addressing may be found in 6.6.

5.6 Service Conventions

This part of ISO/IEC 9506 uses the descriptive conventions contained in the OSI Service Conventions (ISO/TR 8509). The OSI Service Conventions define the interactions between the MMS-user and the MMS-provider. Information is passed between the MMS-user and the MMS-provider by service primitives, which may convey parameters. The following apply to the use of this model:

a) ISO/TR 8509 defines a model for the service provided by a layer of the OSI Reference Model. The MMS service does not correspond to such a layer (it describes a part of the application layer) but the model used is identical in all other respects;

b) at any instant in time, an application entity has multiple service requests outstanding, each proceeding independently of the others.

NOTE – It should be noted that the MMS-user/MMS-provider distinction is an abstraction, and may not necessarily correspond to the realization of MMS in any particular system. Clauses 6 and 7 provide further details on the usage of abstract models.

5.7 Calling and Called MMS-user

This part of ISO/IEC 9506 makes use of the terms Calling and Called MMS-users. The Calling MMS-user is the MMS-user that issues the Initiate.request service primitive. The Called MMS-user is the MMS-user that issues the Initiate.response service primitive.

NOTE – The use of the term "called" in MMS is not the same as the general usage of the term in OSI. The MMS usage of the term "called" corresponds to the OSI usage of the term "responding". This distinction has been introduced in order to avoid confusion with the Requesting/Responding MMS-user definition given below.

5.8 Sending and Receiving MMS-user and MMPM

This part of ISO/IEC 9506 makes use of the terms Sending and Receiving MMS-users. The Sending MMS-user is the MMS-user that issues a request or response service primitive. The Receiving MMS-user is the MMS-user that receives an indication or confirmation service primitive.

NOTE – It is important to note that, in the course of completion of a confirmed MMS service, both MMS-users will be senders and receivers at one time. The first MMS-user sends the request and receives the confirmation, while the second MMS-user receives the indication and sends the response.

This part of ISO/IEC 9506 makes use of the terms Sending and Receiving MMPMs. The Sending MMPM is the MMPM that sends an MMS PDU. The Receiving MMPM is the MMPM that receives an MMS PDU.

5.9 Requesting and Responding MMS-user

This part of ISO/IEC 9506 makes use of the terms Requesting and Responding MMS-users. The Requesting MMS-user is the MMS-user that issues the request service primitive for a service, while the Responding MMS-user is the MMS-user that issues the response service primitive for a service.

NOTE – It is important to note that the use of the term Responding MMS-user differs from the use of the term Responding entity in ACSE and other Standards. In those Standards, the term is used to reference the entity that responds to a connection request.

5.10 Client and Server of a Service

This part of ISO/IEC 9506 makes use of the terms Client and Server in order to describe the model of the MMS VMD (The VMD is described in clause 7). The Server is defined as the peer communicating entity which behaves as a VMD for a particular service request instance. The Client is the peer communicating entity which makes use of the VMD for some particular purpose via a service request instance. The VMD model is primarily useful in describing the actions of the Server, and thus in describing the commands and responses that a Client may use. A real end system may adopt the Client role, or the Server role, or both during the lifetime of an application association. Use of MMS in the OSI environment is further described in clause 6.

Figure 1 depicts the relationships of the client and server of a service, the requesting and responding MMS-user, and the sending and receiving MMS-user and MMPM.

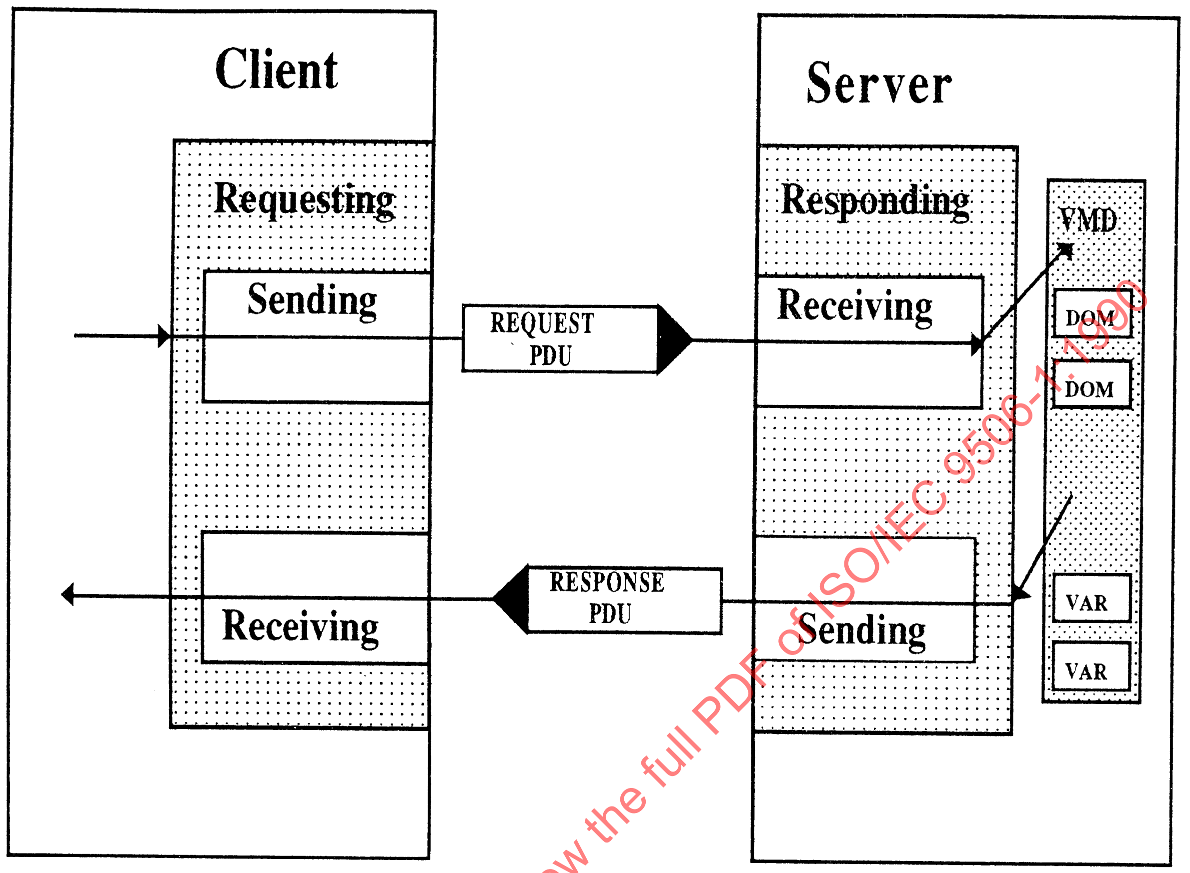


Figure 1 – Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MMPM

5.11 Object modelling

This part of ISO/IEC 9506 makes use of a technique of abstract object modelling in order to fully describe the MMS device model and the MMS service procedures. In this modelling technique, abstract objects, the characteristics of such objects, and operations on those objects are described. The objects defined are abstract and aid in the understanding of the intent of MMS service procedures and their effects. In implementing MMS, a real system maps the concepts described in the model to the real device. Hence, as viewed externally, a device that conforms to this part of ISO/IEC 9506 exhibits the characteristics described in the object modelling technique, but the mechanisms for realization of this view are not defined by this part of ISO/IEC 9506.

MMS defines a number of classes of objects. Each object is an instance of a class, and constitutes an abstract entity which exhibits certain characteristics and may be affected by certain MMS services and operations. Each class is given a name, by which it may be referenced.

Each class is characterized by a number of attribute types, which serve to describe some externally visible feature(s) of all objects of this class. Each instance of a class (object) has the same set of attribute types, but has its own set of attribute values. The values of these attributes are defined by this part of ISO/IEC 9506 or may be established by MMS services, and hence their effect on the device may be modeled by a change in one or more attribute values of an object (or objects).

Each object must be uniquely identified among all instances of the same class. For this purpose, one or more of the object's attribute values, as a combination, must be unique. (For example, many objects have an attribute type called "object name", which is different for each object of the same class.) In MMS, each attribute which is a part of this combination of attributes which make the object unique is identified as a "key attribute".

Finally, some objects contain attributes which are conditional, in the sense that they are relevant to the object if and only if certain conditions hold true. MMS expresses such attributes through the use of a "constraint", which specifies a condition. Attributes that are subject to a constraint are considered to be object attributes for an object if and only if the corresponding constraint is satisfied for that object.

In MMS, classes are syntactically defined as a set of objects as follows:

```
Object: (name of class)
    Key Attribute: (name of attribute type (values))
    .
    .
    .
    Key Attribute: (name of attribute type (values))
    Attribute: (name of attribute type (values))
    .
    .
    .
    Attribute: (name of attribute type (values))
    Constraint: (constraint expression)
        Attribute: (name of attribute type (values))
        Attribute: (name of attribute type (values))
        Attribute: (name of attribute type (values))
```

By convention, each object definition begins with an object declaration and the name of the object. Immediately following, and indented, one or more Key Attributes are named. Next, zero or more Attributes are named. Note that constraints may be expressed anywhere within the attributes, with the convention that all attributes subject to the constraint are indented underneath it. The first attribute definition that is not indented ends the list of attributes that are subject to the constraint.

NOTE – For convenience, attributes may be indented under other attributes in order to show a hierarchy of nesting of such attributes.

5.12 References to Objects

Some objects contain attributes which make reference to other objects. Such attributes, called reference attributes, provide a mechanism to create a linkage from one object to another. The values used to represent such attributes in a real system is a local matter, and such attributes may not be directly modified or examined. Many MMS services provide the capability to determine the identity of an object referenced by such a linkage, however, through the use of such an indirect reference.

NOTE – Reference attributes are similar to "pointer" types available in many programming languages. As an example of the operation of reference attributes, the MMS Rename service may be used to change the Object Name attribute of a referenced object, while having no effect on the value of the reference attribute.

Reference attributes may take on the value UNDEFINED when the object referenced by the link is deleted. In this case, the consequences of this value and its use on the behaviour on other objects is specified in the relevant object and service descriptions. Once a reference attribute becomes UNDEFINED, creation of a object of the same type as the deleted (and referenced object) with the same attribute values as that object which was deleted does not change the reference attribute (it retains the value UNDEFINED).

5.13 Parameter Types

Types for parameters defined in this part ISO/IEC 9506 make use of types defined in ISO 8824, clause 3. Additionally, complex types are used, which are constructed from the ISO 8824 primitive types and subsequently named to allow them to be referenced.

6 MMS in the OSI Environment

This clause serves to describe the relationship between MMS and the OSI environment, and relates OSI terms and concepts to the manufacturing environment. Clause 7 describes the specific model of the MMS device within this environment.

The development of standards for the interconnection of manufacturing monitoring and control devices is assisted by the use of abstract models. To specify the externally perceived behaviour of interconnected manufacturing devices, each manufacturing device is represented by a functionally equivalent abstract model of the device called a Virtual Manufacturing Device (VMD) (see clause 7). The VMD model, along with the extensions to this model which are provided by clauses 8 to 16, describes the externally visible aspects of these devices.

In order to accomplish this, however, it is necessary to describe both the internal and external behaviour of these manufacturing devices. Only the external behaviour of the devices is retained as the standard of behaviour of a VMD. The description of the internal behaviour of such devices is provided in the model only to support the definition of the externally perceived aspects. Any manufacturing device which behaves externally as a VMD can be considered to be in conformance with ISO/IEC 9506.

NOTE 1 – The reader not familiar with the technique of abstract modelling is cautioned that the concepts introduced in the description of the VMD constitute an abstraction despite a similar appearance to concepts commonly found in real devices. Therefore, the VMD model is not a specification for an implementation.

Additional information to supplement this clause may be found in the OSI Reference Model (ISO 7498-1 and ISO 7498-3), the OSI Application Layer Structure (ISO 9545), and the Service Definition for the Association Control Service Element (ISO 8649). References may be found in this part of ISO/IEC 9506, clause 2. This information may be particularly useful in understanding the relationship between AEs, APs, and their respective invocations.

NOTE 2 – This part of ISO/IEC 9506 specifies MMS operation in the OSI environment only. Operation in other environments and architectures is beyond the scope of this part of ISO/IEC 9506, but may be standardized elsewhere.

6.1 Information Processing Tasks and Real Systems

The control and monitoring of a manufacturing process is a distributed information processing task. In order to carry out successfully this task, inter-operation of a number of real open systems is required. In OSI, a real open system is defined as a set of computers and associated software (including peripherals, terminals, human operators, physical processes, etc.) that complies with the requirements of OSI standards in its communications with other real systems.

In OSI, the inter-operation of real open systems is modelled in terms of the interactions between Application Processes (APs) in these systems. The distributed task of control and monitoring of a manufacturing process requires the co-operation of two or more Application Processes.

6.2 Application Processes

An Application Process (AP) is an element within a real open system which takes part in the execution of one or more information processing tasks. It is an abstract representation of those aspects of a real open system which are specific to the performance of those tasks. APs generally have requirements above and beyond any requirements to communicate with other APs. In the manufacturing environment, an AP represents a real system participating in the overall control/monitoring distributed task.

6.3 Interaction of Application Processes

Several requirements must be met in order to allow co-operative operation of APs. First, they must share sufficient information to enable them to interact and carry out processing functions in a compatible and co-operative manner. In the manufacturing environment, the term "universe of discourse" is used to name a model of those aspects of the "world" which are pertinent to the objectives of a manufacturing control and monitoring task of an AP. In order to allow successful interworking between APs, they must share a universe of discourse (i.e. they must have a common understanding of the manufacturing environment).

It is convenient to describe the common model of the manufacturing environment, or universe of discourse, in terms of a set of abstract "objects". Examples of objects in the manufacturing environment include variables, programs, and semaphores. The model describes the characteristics of objects and relationships between them. The characteristics may include the properties of those objects (either static or dynamic), and these properties are expressed as a set of rules and constraints about the behaviour of these objects within the model of the manufacturing environment. An example of a property of an object is the characteristic of a program that it is either running or stopped.

A universe of discourse can be described formally by a "conceptual schema". The conceptual schema for the manufacturing universe of discourse is described by the models provided by MMS. The second requirement for successful interworking of two APs is that they share a common model of the objects (such as variables, programs, and semaphores) in the universe of discourse. In OSI, this is called a "shared conceptual schema". This requirement is met in a manufacturing control and monitoring application through the sharing of the models in MMS for the manufacturing universe of discourse.

When two APs co-operate, their behaviour is determined partly by these models (the shared conceptual schema) and partly by their past interactions. The past interactions are modelled by the state of the objects in the universe of discourse. As an example, the co-operative behaviour of two APs may be partly determined by the Program Invocation model of MMS and partly by the state of Program Invocation objects in the APs. The shared information about the state of objects is called an information base.

6.4 Interaction of Application Processes in OSI

The activity of a given AP in a specific information processing task is supported by an application-process-invocation. At any time, an Application Process may have zero, one or more application-process-invocations. While the Application Process describes a specific set of information processing functions in a particular real system, the application-process-invocation (AP-invocation) describes an instance of the Application Process in a real system for a particular occasion of information processing. (Hence, an AP may describe control of a robot arm in a specific real system, while an AP-invocation describes the control of a robot arm in a specific real system on some occasion of control for assembly of some particular part).

Thus, co-operation between APs for the performance of a given information processing task takes place through interacting AP-invocations. When APs interact using OSI communication capabilities, these communication capabilities are in turn modelled by one or more Application Entities (AE). An AE is an active element in the Application Layer of an open system, and it represents those parts of an Application Process involved in OSI communications. Each AE represents exactly one AP.

Like APs, the activity of a given AE is supported by an application-entity-invocation. An AE-invocation performs the functions of an AE for a particular occasion of communication. At any time, an AP-invocation may be represented by zero or more AE-invocations for each AE associated with the AP. Hence, the interactions of AP-invocations for a particular occasion of communication is represented in OSI by a set of corresponding AE-invocations.

Because an AE describes only part of the operation of an AP (namely, that involving OSI communications), resources in AP-invocations may exist beyond the lifetime of (and may be independent of) an AE-invocation.

NOTE – The relationship between an AE and the VMD is defined in clause 7.

6.5 Structure of Application Entities

The AE represents a set of communication capabilities of an AP. These capabilities are defined by a set of Application Service Elements (ASEs). An ASE is a coherent set of integrated functions that provides a capability for communications for some particular purpose. MMS is modelled as an ASE for the purpose of communications with manufacturing devices. Support for the Association Control Service Element (ACSE) ASE, as defined in ISO 8649, is required in every AE that supports the MMS ASE.

6.6 Addressing of Application Entities

ISO 7498-3 describes naming and addressing concepts in the OSI environment. This subclause describes how some of these concepts apply to MMS.

A Presentation-address is associated with a set of PSAPs in a single real system. An AE is attached to one or more Presentation Service Access Points (PSAPs) in order to make it addressable in the OSI environment.

Each AE is identified by one or more application-entity-titles (AE-titles), which are unambiguous throughout the OSI environment (OSIE). At any instant of time, each AE-title is bound to a single Presentation-address that identifies the set of PSAPs to which the AE is attached. This binding is recorded in the Application Title Directory Service, which contains information about AEs.

A system may access the Application Title Directory Service via the OSI Directory Service (ISO 9594). Conceptually, each system contains a directory function, from which AEs may obtain addressing information. This directory function may operate via the use of a local cache, or may utilize the Application Title Directory Service via the OSI Directory Service, in order to satisfy requests from AEs.

NOTE – This is only a conceptual model, and that there are many valid implementation styles.

In order for an application-entity to establish an application association (AA) with another AE, it must know the Presentation-address or obtain it by using the called AE-title to get the Presentation-address of the AE from the directory function. It then uses this Presentation-address to establish a presentation connection with the called AE. The mechanisms for control of associations are provided in ISO 8649 and ISO 8650 by the Association Control Service Element (ACSE). Since an AE is the representation of an AP within OSI, the establishment of communication to support an association with the called AE is the establishment of communication with the related AP.

MMS makes use of an "Application Reference" to identify an AE in another system, by making use of its title as defined in ISO 8649. The syntax of an Application Reference is defined in ISO/IEC 9506-2.

6.7 Application Context

The application context, which is established via the Association Control Service Element (ACSE) (ISO 8649), identifies the set of application service elements, their options, rules for use of the service elements, and their effects that are available on an application association. In the case of MMS, the application context associated with MMS indicates that the rules and requirements associated with ISO/IEC 9506 are in effect when the MMS application context is negotiated.

6.8 Presentation Context, Abstract Syntaxes, and Transfer Syntaxes

In OSI, there is an important distinction between the generic requirements of an application for the transfer of data and how those requirements are met in terms of a specific representation of data values. The former aspect of data description is referred to by the term "abstract syntax", while the latter aspect is referred to by the term "transfer syntax".

Abstract syntaxes are intimately associated with application protocol standards. ISO/IEC 9506-2 defines an abstract syntax matching its data transfer requirements. An abstract syntax can be viewed informally as describing the generic structure of data. In MMS, the set of type definitions provided in ISO/IEC 9506-2 constitutes an abstract syntax. The MMS abstract syntax may be supported by many different transfer syntaxes.

Transfer syntaxes are concerned with the way in which data is actually represented in terms of bit patterns during transmission. A transfer syntax may have attributes that are not related to the abstract syntaxes which it can support, but such attributes may be significant. For example, a transfer syntax may provide data compression or encryption. Such attributes (and how well they meet the requirements of the device) may influence the choice of syntaxes offered or selected for an instance of communication.

For the purpose of transferring data between MMS entities, it is necessary to identify the abstract syntax being used (MMS) and a transfer syntax that is capable of representing data values that are generated using this abstract syntax. The presentation service (ISO 8822) provides the mechanism by which appropriate abstract and transfer syntaxes are negotiated. A specific combination of an abstract syntax and a transfer syntax that is used for transfer of data using the presentation service is called a presentation context.

Abstract Syntax Notation One (ASN.1) is an example of a tool for specification of syntaxes and associated encoding rules. (The application of encoding rules to a particular abstract syntax may generate a transfer syntax.) The ASN.1 Specification (ISO 8824) has been chosen to describe the MMS abstract syntax. One possible transfer syntax for MMS is defined by the application of the ASN.1 Encoding Rules (ISO 8825) to the MMS abstract syntax. Other transfer syntaxes may be negotiated via the presentation layer. ISO/IEC 9506 requires that any system claiming conformance to ISO/IEC 9506 shall support the transfer syntax that results from the application of the ASN.1 Basic Encoding Rules (ISO 8825) to the MMS abstract syntax. Support of other transfer syntaxes is a local matter. See ISO/IEC 9506-2, clause 18.

7 The Virtual Manufacturing Device

7.1 Introduction

The MMS services define the externally visible behaviour of an MMS server application process. This behaviour is modelled by describing an entity called a Virtual Manufacturing Device (VMD). This clause describes the model for a VMD. It explains the VMD's relationship to the OSI application process and the OSI application entity (AE). It defines the structural elements of the VMD and introduces the abstract objects which exist at, and are manipulated by, a VMD on behalf of a client MMS-user.

An implementation of an MMS server must provide a mapping of the VMD model on to the functionality of a real manufacturing device. Guidance in the selection of a particular mapping may be found in various Companion Standards. These Companion Standards address the specific needs of discrete parts manufacturing systems - numerical controllers, programmable controllers, robotic controllers, and vision systems - and of batch and continuous process control systems.

NOTE – The MMS services do not constrain the behaviour of a client MMS application process, except with respect to valid sequences of primitives. Therefore a model of the MMS client application process is not given.

7.1.1 Relationship of the VMD to the OSI Model

A VMD exists within the MMS server application process. It constitutes that portion of an information processing task which makes available - for control, or monitoring, or both - a set of resources and functionality associated with a real manufacturing device. An application process may contain zero or more VMDs. If it does not define a VMD it may not act as an MMS-server.

Each VMD represents a virtual manufacturing device within that AP, and each VMD is logically separate from all other VMDs.

EXAMPLE – An MMS system which is connected to a non-MMS environment containing multiple attached manufacturing devices could be modelled as a single application process containing one VMD for each attached device or as several application processes, each containing a single VMD for a single, distinct, attached device. The clients of the VMDs in either case will see a particular attached device as a single VMD. Relative to MMS services, this VMD will appear to be independent of all other VMDs.

Each AP may contain zero or more AEs, such that an AE represents a set of communication capabilities of the AP (an AP that contains no AEs may not communicate in the OSI environment).

A VMD may wholly contain zero or more AEs. Each AE in a VMD represents a set of communication capabilities used by the aspects of the AP represented by a VMD. Each AE is related to one and only one VMD. When a VMD contains more than one AE, then it contains more than one set of communication capabilities.

At any instant of time, each AE-title is bound to a single presentation-address that identifies a set of PSAPs to which the AE-title is bound and therefore the AE-title is addressable in OSI. Communication with a particular AE is used to model communication with a VMD, and hence an AP. Application associations are modelled as taking place between AEs (and hence VMDs and APs).

An AE within an AP may have zero or more AE-invocations. For the purposes of MMS, each AE-invocation models an instance of usage of that AE in an application association. Hence, several AE-invocations of an AE may be used to model several application associations with a VMD that contains that AE. By generalization, there may simultaneously exist several AEs of a VMD, with several associations each (modelled by multiple AE-invocations). Note that in OSI a more generalized view allows an AE-invocation to have more than one application association.

An OSI Presentation Address which identifies one or more PSAPs addresses a single VMD. This binding of a Presentation Address to a VMD is of a relatively long duration. Figure 2 illustrates the relationship between an OSI application process acting as an MMS server, its VMDs, and the PSAPs used to access them.

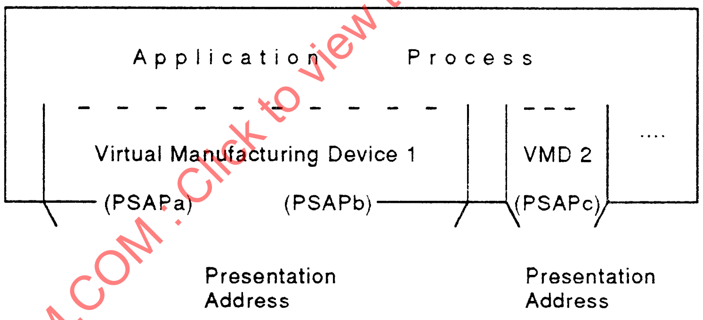


Figure 2 – The MMS Server Application Process

7.1.2 Relationship of the VMD to the Real Manufacturing Device

A VMD is an abstract representation of a specific set of resources and functionality at the real manufacturing device and a mapping of this abstract representation to the physical and functional aspects of the real manufacturing device. This mapping of a virtual resource to the underlying actual resource is of relatively long duration.

Generally, the resources of a given VMD are distinct from, and independent of, the resources of all other VMDs. When a virtual resource of two (or more) VMDs is mapped to the same underlying physical resource, a mechanism must be provided by the application process(es), and made available through the VMDs, so that clients of the various VMDs may coordinate their access to the single real resource. This coordination may be modelled by requiring each VMD to obtain control of a "virtual" semaphore whenever disruptive access to the virtual resource is being requested by an MMS service.

This virtual semaphore would then be mapped on to a real semaphore which controls access to the real resource. With this approach, the virtual semaphores of the various VMDs are independent when viewed using MMS services. In other words, users of the virtual semaphore of one VMD appear as though part of the resident system when viewed by users of the semaphores of the other VMDs. (Semaphores are described in clause 13.)

MMS describes the operation of a VMD by describing the abstract objects which are manipulated by it, and by describing the set of operations which may be performed on these objects through use of the MMS services.

7.2 The Structure of a VMD

Each VMD contains exactly one Executive Function and zero or more Program Invocations, each of which depend on one or more Domains. The Domain represents a specific use of a set of capabilities of the VMD. The state of the VMD, in the complete sense, is determined by the values of all the attributes of the VMD, including all the attributes of its Domains and their subordinate objects. The elements of the VMD are described below.

The VMD Object Model
Object: VMD

```
Key Attribute: Executive Function
Attribute: Vendor Name
Attribute: Model Name
Attribute: Revision
Attribute: List Of Abstract Syntaxes Supported
Attribute: Logical Status (STATE-CHANGES-ALLOWED,
            NO-STATE-CHANGES-ALLOWED, LIMITED-SERVICES-PERMITTED,
            SUPPORT-SERVICES-ALLOWED)
Attribute: List Of Capabilities
Attribute: Physical Status (OPERATIONAL, PARTIALLY-OPERATIONAL,
            INOPERABLE, NEEDS-COMMISSIONING)
Attribute: List Of Program Invocations
Attribute: List Of Domains
Attribute: List Of Transaction Objects
Attribute: List Of Upload State Machines (ULSM)
Attribute: Lists of Other VMD-specific Objects
Attribute: Additional Detail
```

7.2.1 The Executive Function

The existence of a fully functional Executive Function exactly corresponds with the existence of the VMD.

7.2.2 Vendor Name

This attribute is a character string which identifies the vendor of the system which supports this VMD.

7.2.3 Model Name

This attribute is a character string which identifies the model of the system which supports this VMD. The value of this string is normally assigned by the vendor.

7.2.4 Revision

This attribute is a character string which identifies the revision level of the system which supports this VMD. The value of this string is normally assigned by the vendor.

7.2.5 List Of Abstract Syntaxes Supported

This attribute identifies the set of Abstract Syntaxes which this VMD is able to support in the MMS Application Context. This set includes any abstract syntaxes defined by a Companion Standard as well as any abstract syntax which can be recognized as an encoding for Load Data (see 10.3) and Execution Argument (see 11.4). The abstract syntax defined in ISO/IEC 9506-2, clause 19, shall not be included in the list.

7.2.6 Logical Status

There are four separate levels of functionality available through MMS which are described by the Logical Status attribute of the Executive Function.

7.2.6.1 STATE-CHANGES-ALLOWED

In this case, all MMS services which are supported by this VMD may be performed.

7.2.6.2 NO-STATE-CHANGES-ALLOWED

In this case, the only MMS services which may be performed (if the VMD supports the service) are:

Abort
Conclude
Cancel
GetAlarmEnrollmentSummary
GetAlarmSummary
GetCapabilityList
GetDomainAttributes
GetEventActionAttributes
GetEventConditionAttributes
GetEventEnrollmentAttributes
GetNamedTypeAttributes
GetNamedVariableListAttributes
GetNameList
GetProgramInvocationAttributes
GetScatteredAccessAttributes
GetVariableAccessAttributes
Identify
Initiate
Read
ReadJournal
ReportEventActionStatus
ReportEventConditionStatus
ReportEventEnrollmentStatus
ReportJournalStatus
ReportPoolSemaphoreStatus
ReportSemaphoreEntryStatus
ReportSemaphoreStatus
Status

7.2.6.3 LIMITED-SERVICES-PERMITTED

In this case, the only MMS services which may be performed are: Abort, Conclude, Status and Identify.

7.2.6.4 SUPPORT-SERVICES-ALLOWED

In this case, all MMS services supported by the VMD except the Start, Stop, Reset, Resume, and Kill services may be performed.

7.2.7 List Of Capabilities

A VMD represents the real device to the client by providing a set of capabilities which may be used by the client in order to effect some control or monitoring (or both) activity through the VMD.

A capability is a locally defined resource (physical or logical) or a locally defined set of resources which can be identified by a character string. The definition and management of capabilities is outside the scope of this part of ISO/IEC 9506 However, it is assumed that the existence and status of capabilities is known by the Executive Function, and that the Executive Function contains sufficient knowledge in order to "allocate" capabilities to the various Domains which may be created. No object model is provided for a capability, since it is considered primitive from the MMS point of view.

A capability may be sharable (or not) depending on local criteria. Additionally, a capability may be distinct, or it may encompass (or be encompassed by) one or more other capabilities.

NOTE – An implementation of an MMS server should provide a mapping of the VMD model on to the functionality of a real manufacturing device. Guidance in the selection of a particular mapping may be found in various Companion Standards. In particular, the Companion Standards will aid in the identification of Domains and Program Invocations with features of real systems. The attribute "capability" is intended to allow expression of features of the implementation needed for relating any real implementation to the VMD model and to the extended model provided by the Companion Standard. The use of capabilities in any MMS implementation requires prior agreement between implementations of MMS client and server.

EXAMPLE 1 – In a programmable controller application, a capability "BlockMem1" may be assigned to represent a collection of memory address ranges which are available for the purpose of customizing a standard control program for the manufacture of a specific part.

EXAMPLE 2 – In a robotics application, the capability "Arm1" may represent the set of sensors, actuators, and logic associated with a specific arm of a multi-arm robot.

7.2.8 Physical Status

Associated with each capability is one or more attributes which describe the state of the capability. It is outside the scope of part of ISO/IEC 9506 to provide a uniform representation of these attributes. However, it is useful to provide a standard representation of the gross aspects of all the capabilities, taken together, in order to characterize the operational state of the hardware. The Physical Status attribute (of the entire collection of capabilities) represents this attribute of the real device.

NOTE – This status refers to the hardware associated with the device's operation. It is unrelated to the ability of the device to communicate in an OSI environment.

7.2.8.1 OPERATIONAL

The real device associated with this VMD has no known deficiencies and is able to perform its intended tasks.

7.2.8.2 PARTIALLY-OPERATIONAL

One or more functions of the real device cannot be performed due to hardware malfunctions or limitations.

7.2.8.3 INOPERABLE

One or more significant problems exist at the real device which prevent it from doing any useful task.

7.2.8.4 NEEDS-COMMISSIONING

The device is in a state such that a local commissioning process needs to be performed before useful tasks can be accomplished.

7.2.9 List Of Program Invocations

A Program Invocation consists of a set of procedural and data elements contained within Domains together with execution control information. These elements may be predefined within the VMD, or they may be dynamically defined (for example by creation of a Domain through use of the MMS load services), or both. The Program Invocation itself may be predefined within the VMD, or it may be dynamically created and deleted either by local means or through the use of MMS services. An object model of the Program Invocation is given in clause 11.

7.2.10 List Of Domains

A Domain represents a specific instance of use of a set of capabilities of the VMD. A Domain includes those aspects of a VMD which are associated with a specific element (possibly all) of a coordinated control or monitoring (or both) strategy. The allocation of the capabilities of a VMD to the Domains may be static or it may be dynamic. If the allocation is static, then the Domain is predefined within the MMS server and its name is considered to be known. If the allocation is dynamic, the Domain comes into existence and is removed from the VMD either through the action of MMS services or through local actions. Within a given VMD, either or both types of Domains may exist. Domains are further described and an object model provided in clause 10.

A Domain may be empty or it may contain "information". The "information" may be program instructions for some processor or tables of values or other classes of data or all of the above. For dynamic Domains which come into existence through MMS services, the Domains are implicitly created by the process of loading their contents. Dynamic Domains may also be created through actions of the Program Invocation when it is in execution or through other local means.

The Domain content is often closely associated with a file. The concept of file, although it appears in some MMS services, is not defined within MMS. The file may be part of a virtual Filestore (ISO 8571) which is part of the Application Process which contains the VMD, or it may be locally defined. If the "information" within a file is a part of the VMD and can be used directly by the VMD (as a component of a Program Invocation), it can be considered to be a Domain. The file, however, must be considered a separate object, outside the scope of the VMD.

EXAMPLE – A numerical control application may require several Domains. One Domain may consist of the resources associated with part program execution and other Domains may be associated with one or more material handling functions. A third static Domain may be associated with a predefined calibration procedure. Thus Domains not only subset the physical resources - such as memory and I/O devices - of the numerical controller, but they also partition the functionality of the controller.

Most MMS objects are defined subordinate to (within the name space of) a Domain. Thus, a Domain represents a single name space in which MMS objects must be uniquely identifiable.

7.2.11 Transaction Object

Most MMS services are confirmed services (see ISO/IEC 9506-2, clause 6). When the VMD receives an indication primitive for one of the confirmed services, a Transaction Object is created which governs the processing of this service. The description of that object follows:

Object: Transaction

```
Key Attribute: Invoke ID
Key Attribute: Application Association Identifier
Attribute: List Of Pre-execution Modifiers
Attribute: Current Modifier Reference
Attribute: Confirmed Service Request
Attribute: List Of Post-execution Modifiers
Attribute: Cancelable (TRUE, FALSE)
```

Invoke ID

This attribute is an integer which serves to identify the transaction within the Application Association.

Application Association Identifier

This attribute identifies the application association over which this transaction object is created.

List Of Pre-execution Modifiers

An ordered list which identifies modifiers which must be satisfied before execution of the Confirmed Service Request attribute can begin.

Current Modifier Reference

References the Semaphore Entry object or the Event Enrollment object controlling the current modifier's execution.

Confirmed Service Request

Service identifier and Argument of the pending service.

List Of Post-execution Modifiers

References the Semaphore Entry objects which this service invocation owns due to a processed Attach to Semaphore modifier.

Cancelable

Initially true, the service may set this attribute false (local issues). When true, cancel works; when false, cancel does not work.

7.2.11.1 Initialization of Transaction Objects

A transaction object shall be created upon receipt of an indication service primitive for an MMS confirmed service, and deleted after the MMS-user issues a response service primitive for that service instance. The number of transaction objects which may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

When an indication service primitive is received, the List Of Pre-execution Modifiers attribute of the newly created transaction object shall be set based on the List Of Modifier parameter on the indication service primitive (see 5.4). The Current Modifier Reference is set to one, indicating the first Modifier on the List Of Pre-execution Modifiers, or to zero if there are no Pre-execution Modifiers. The List Of Post-execution Modifiers is set to empty (no modifiers are on this list).

The Confirmed Service Request attribute of the transaction object shall be set based on the Argument parameter received on the indication service primitive.

The Cancelable attribute shall initially be set to true. The Cancelable attribute of the transaction object shall not be set to false during the processing of the pre-execution modifiers. After all pre-execution modifiers have been processed, the cancelable attribute may be set to false by the MMS-user at any time as a local matter, subject to the requirements for the Cancel service and the service named in the Confirmed Service Request. The cancelable attribute shall have been set to false by the time that post-execution modifier processing takes place.

7.2.11.2 Processing of Transaction Objects

After a transaction object has been initialized by the MMS-user, processing may begin on that object. Processing begins with the List Of Pre-execution Modifiers, followed by execution of the Confirmed Service Request, and terminates with processing of the List Of Post-execution modifiers.

NOTE – Although this part of ISO/IEC 9506 requires that Transaction Objects be initialized in the order of receipt, there is no such requirement for the order of processing of these objects. To enforce serialization of actions, the MMS client should wait for a service response before issuing subsequent service requests.

Each of the Pre-execution modifiers shall be processed in order. Each modifier shall complete successfully before the next modifier can be processed. If a modifier fails (see definition of particular modifiers), the MMS-user shall process the Post-execution modifier list (as specified below) and then issues a response(-) service primitive, specifying the appropriate error class and code, and the transaction object shall be deleted. The Current Modifier Reference shall be set to indicate the modifier currently being processed in the list. The Current Modifier Reference shall be identified by an integer, where 1 indicates the first element in the list. For each AttachToSemaphore modifier which is successfully processed, an entry shall be made in the List Of Post-execution Modifier attribute (in order to allow relinquishing of control of the semaphores after service execution). The List Of Post-execution Modifier shall be constructed in reverse order, such that the first pre-execution modifier becomes the last post-execution modifier.

After all pre-execution modifiers have been processed successfully (if any are specified on the indication service primitive), the Confirmed Service Request shall be executed in accordance with the service procedure specified in this part of ISO/IEC 9506 for the named service.

Following completion of the service request, the MMS-user shall process the post-modifiers in the List Of Post-execution Modifiers attribute in the order that they are specified in the list (note that this is the opposite of the order in which the pre-modifiers were executed).

After all post-execution modifiers have been processed, the response service primitive shall be issued by the MMS-user. The Current Modifier Reference attribute is used to indicate the current modifier during the processing of this list.

7.2.12 Upload State Machines

The Upload State Machine object (ULSM) is created through the InitiateUploadSequence service which is described in clause 10.

7.2.13 Other VMD-specific Named Objects

There are many other named objects which may have VMD-specific scope. The attributes of such objects shall be considered part of the VMD. General considerations for all named objects are described in 7.3, and the objects themselves are described in separate clauses of this part of ISO/IEC 9506.

7.2.14 Additional Detail

This attribute contains zero or more attributes whose meaning and syntax are defined by appropriate Companion Standards.

7.3 Specification of Named Objects

MMS objects are normally referenced by name. The name of an object shall be unique within its scope of definition and within the class of object it identifies. In MMS, the names of objects are defined within one of three scopes: within a VMD instance, within a Domain, or within a single application association.

7.3.1 Scope of Names

In general, MMS names can have one of three scopes, VMD-specific Domain-specific, and Application Association-specific.

7.3.1.1 VMD-specific Scope

A name which has VMD-specific scope shall be unique among all VMD-specific objects of the same object class. Such a name may be referenced by all clients of the VMD instance on any application association with it. VMD-specific objects are not automatically deleted when the MMS application association ceases to exist.

7.3.1.2 Domain-specific Scope

A Domain represents a single flat name space. A name which is Domain-specific shall be unique for its class of object within the Domain in which it is defined. The unique identification of such an object requires the specification of the name of the Domain and the name of the object, thus implying a two level hierarchy.

7.3.1.3 Application Association-specific Scope

A name having "Application Association-specific" (AA-specific) scope may only be referenced by the client MMS-user for which the name was defined, and only on the specific application association over which the name's definition is valid. The definition of an object bearing an application association-specific name, if not explicitly deleted previously, shall be deleted whenever the defining application association ceases to exist.

7.3.2 Classes of Objects

The specific instances of MMS objects which can be named are listed in Table 1. Not all objects can have all possible Name Scopes. The allowed combinations are indicated by Xs in Table 1.

Table 1 – Name Class and Scope

	VMD	Domain	AA	clause
Named Variable Objects	X	X	X	12
ScatteredAccess Objects	X	X	X	12
Named Variable List Objects	X	X	X	12
Named Type Objects	X	X	X	12
Semaphore Objects	X	X		13
Event Condition Objects	X	X	X	15
Event Action Objects	X	X	X	15
Event Enrollment Objects	X	X	X	15
Journal Objects	X		X	16
Domain Objects	X			10
Program Invocation Objects	X			11
Operator Station Objects	X			14

In addition to the requirement that the name of each object be unique within its Name Scope and for the class of the object, there is an additional requirement on the first two objects listed. Named Variable Objects and Scattered Access Objects share a common name space. Therefore the name of any such object shall be unique within this common name space for its Name Scope.

NOTE – There are other limitations on the names of Event Conditions, Event Actions, and Event Enrollments. Creation of a semaphonre automatically creates an Event Condition of the same name (see 13.4). Therefore, a semaphore cannot be proposed for creation which has the same name as an existing Event Condition. If the monitor parameter is selected, the creation of a Program Invocation automatically creates an Event Condition, and Event Action, and an Event Enrollment, all having the same name as the Program Invocation (see 11.2). Therefore, a Program Invocation cannot be proposed for creation whose name is the same as an existing Event Condition, Event Action, or Event Enrollment.

7.3.3 Object Lifetime

Each of the objects listed above has a lifetime within the VMD which can be inferred from the scope of its name.

VMD-specific Scope
Such objects exist as long as the VMD exists (unless explicitly deleted).

Domain-specific Scope
Such objects exist as long as the Domain on which they depend exists (unless explicitly deleted).

AA-specific Scope
Such objects exist only as long as the Application Association over which they were defined continues to exist (unless explicitly deleted).

7.3.4 Object Visibility

The Name Scope of a named object also determines the visibility of the object. An object whose name is VMD-specific or Domain-specific may be referenced by any association to the VMD; an object whose name is AA-specific may only be referenced by the association over which the object was defined.

7.3.5 Creation of MMS objects

Each of the MMS objects can either be static, that is predefined within the implementation, or dynamic, coming into existence during the course of operation of the VMD. Static objects are usually not MMS deletable, while dynamic objects are usually MMS deletable, but there may be exceptions to either rule. Static objects are predefined by the implementation and have names assigned to them either by the implementor, or in conformance to this part of ISO/IEC 9506 or to one of the Companion Standards.

Dynamic objects can come into existence either (1) through explicit MMS service procedures, (2) through local action of the system or of the system operators, or (3) through the execution of some Program Invocation. The means of local creation is outside the scope of this part of ISO/IEC 9506. However, if a locally created object is to be visible to MMS services, its external behaviour, including all its visible attributes, shall be consistent with this service definition.

7.3.6 Deletion of MMS Objects

All MMS objects which have the attribute MMS Deletable equal to true may be removed from the VMD through appropriate MMS service requests. In addition to this explicit method of deletion, MMS objects may also be deleted through local action of the system or of the system operators or through the execution of some Program Invocation. The means of local deletion is outside the scope of this part of ISO/IEC 9506. Objects which are subordinate to a Domain are automatically deleted when the Domain is deleted. This is true even if the subordinate object has its MMS-Deletable attribute set to FALSE.

7.3.7 Alteration of MMS Objects

All MMS objects have as part of their description a list of externally visible attributes. In addition to the MMS services which alter these attributes, these attributes may also be changed through the local action of the system or of the system operators, or through the execution of a Program Invocation. In particular, the values of variables associated with Domains or with the VMD directly may change due to the execution of a Program Invocation. The means of local alteration of Object Attributes is outside the scope of this part of ISO/IEC 9506.

A VMD should, if possible, guarantee that access to any MMS object required by an MMS service is not interruptible. In other words, for the entire duration of the MMS service, the object should have a set of attributes that represent the state of the VMD at a single instant of time. Since such guarantees may not be possible, or if possible for some objects may not be possible for all objects, the static conformance statement for the VMD shall state whether uninterruptible access is supported and, if supported, under what constraints it is guaranteed.

7.4 Object Name Structure

The parameter Object Name occurs frequently in the specification of MMS services. The structure of the Object Name is shown in Table 2.

Table 2 – Object Name

Object Name	Req Rsp	Ind Cnf
Name Scope	M	M(=)
AA-specific	S	S(=)
Item Identifier	M	M(=)
Domain-specific	S	S(=)
Domain Identifier	M	M(=)
Item Identifier	M	M(=)
VMD-specific	S	S(=)
Item Identifier	M	M(=)

7.4.1 Name Scope

This parameter specifies which of the possible scopes is to be used for this Object Name.

7.4.2 AA-specific

Some named objects may have an AA-specific name. Such names must be defined over the association on which they exist.

7.4.2.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within this application association for the class of object named.

7.4.3 Domain-specific

Named objects which depend on a Domain may have names which are Domain-specific.

7.4.3.1 Domain Identifier

This parameter, of type Identifier, is itself a VMD-specific name of the Domain which contains the named object.

7.4.3.2 Item Identifier

This parameter, of type Identifier, is the name of the object within the named Domain. This parameter shall be unique within the named Domain for the class of object named.

7.4.4 VMD-specific

Any named object may have a VMD-specific name.

7.4.4.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within the VMD for the class of object named.

7.5 Services on the VMD

ISO/IEC 9506, clause 9, describes the MMS services which operate on the VMD in its entirety. In addition to these services, provision is made for groups defining Companion Standards to define additional services which operate on the VMD.

8 Environment And General Management Services

8.1 Introduction

The environment and general management services contain the Initiate, Conclude, Abort, Cancel and Reject services. These services allow the MMS-user

a) to initiate communication with another MMS-user in the MMS environment, and to establish the requirements and capabilities that support that communication;

b) to conclude communication with another MMS-user in the MMS environment in a graceful manner;

c) to abort communications with another MMS-user in the MMS environment in an abrupt manner;

d) to cancel pending service requests;

e) to receive notification of protocol errors that occur.

8.1.1 Environment Management State Diagram

This clause defines the state diagram for entering and leaving the MMS environment. The initial state for both the calling and called MMS users shall be the state "No MMS Environment". The state diagram is depicted from the point of view of an MMS-user. The state of the MMS environment is within the context of a specific application association. Restrictions on the use of MMS services within this environment shall not apply to other environments established by the MMS-user on different application associations.

8.1.2 Restriction on Use of MMS Services

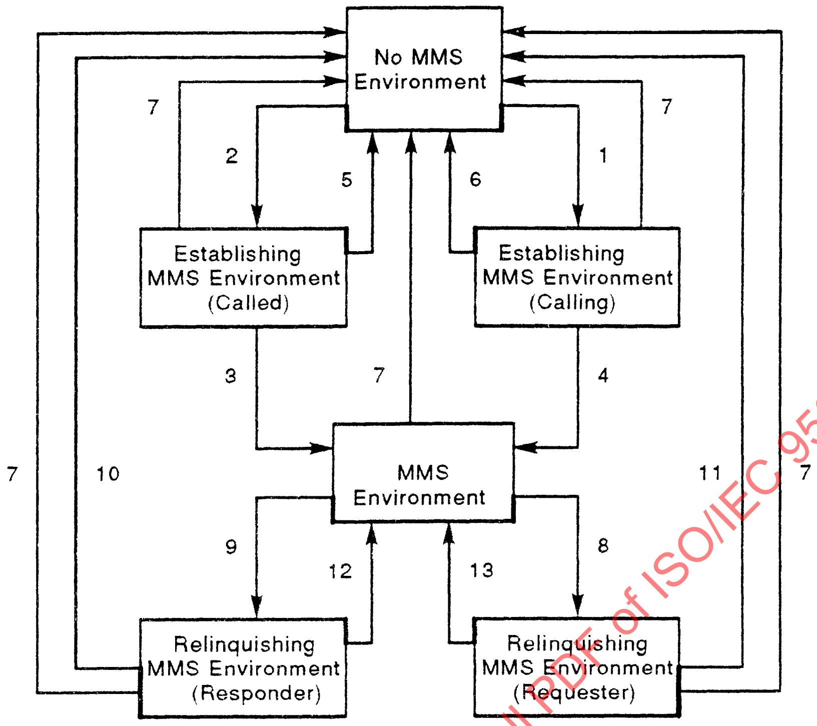
This clause places restrictions on the use of MMS services for certain states. The restrictions in this clause are supplementary to other restrictions placed by other clauses in this part of ISO/IEC 9506.

8.1.2.1 The "No MMS Environment" State

While in the state "No MMS Environment", an MMS-user shall only issue the initiate.request service primitive.

8.1.2.2 The "Establishing MMS Environment (Calling) State

While in the state "Establishing MMS Environment (Calling)", an MMS-user shall only issue the abort.request service primitive.


No MMS Environment
Establishing MMS Environment (Called)
Establishing MMS Environment (Calling)
MMS Environment
Relinquishing MMS Environment (Responder)
Relinquishing MMS Environment (Requester)
1
2
3
4
5
6
7
8
9
10
11
12
13

Transitions:

1 - initiate.request	8 - conclude.request
2 - initiate.indication	9 - conclude.indication
3 - initiate.response +	10 - conclude.response +
4 - initiate.confirm+	11 - conclude.confirm+
5 - initiate.response-	12 - conclude.response-
6 - initiate.confirm-	13 - conclude.confirm-
7 - abort.request or abort.indication	

Figure 3 – Environment Management State Diagram

8.1.2.3 The "Establishing MMS Environment (Called) State

While in the state "Establishing MMS Environment (Called)", an MMS-user shall only issue the initiate.response or abort.request service primitives.

8.1.2.4 The "MMS Environment" State

Clauses 8 to 16 place requirements on the use of service primitives in the state "MMS Environment". This state is divided into a series of substates, which are described in those clauses. All MMS states except the "No MMS Environment", "Establishing MMS Environment (Calling)", "Establishing MMS Environment (Called)", "Relinquishing MMS Environment (Requester)" and "Relinquishing MMS Environment (Responder)" are substates of the "MMS Environment" state.

The only events which cause an exit from the "MMS Environment" state are the issuance of an abort.request, the issuance of a conclude.request service primitive, the receipt of an abort.indication, or the receipt of a conclude.indication service primitive.

While in the state "MMS Environment", an MMS-user may issue any request or response service primitive, subject to the following restrictions:

a) other clauses of this part of ISO/IEC 9506 restrict the usage of services and service primitives via sequencing requirements;

b) a response primitive shall not be issued unless a request primitive for a corresponding service indication has been received (see ISO/IEC 9506, clause 7);

c) the initiate.request service primitive shall not be issued.

8.1.2.5 The "Relinquishing MMS Environment (Requester)" State

While in the state "Relinquishing MMS Environment (Requester)", the only request primitive that the MMS-user may issue is the abort.request service primitive. Note that responses, either (+) or (-) may continue to be issued.

8.1.2.6 The "Relinquishing MMS Environment (Responder)" State

While in the state "Relinquishing MMS Environment (Responder)", the MMS-user may only issue the abort.request and conclude.response service primitives.

8.2 Initiate Service

The Initiate service shall be used to establish the MMS environment and to allow the communicating MMS-users to exchange information regarding their capabilities and requirements. The Initiate service must complete successfully before any other services may be carried out between a pair of MMS-users in the MMS environment.

8.2.1 Structure

The structure of the component service primitives is shown in Table 3.

Table 3 – Initiate Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M			
Local Detail Calling	U	U(=)			
Proposed Max Serv Outstanding Calling	M	M			
Proposed Max Serv Outstanding Called	M	M			
Proposed Data Structure Nesting Level	U	U			
Init Request Detail	M	M			
Result(+) (COMP)			S	S(=)	
Local Detail Called			U	U(=)	
Negotiated Max Serv Outstanding Calling			M	M(=)	
Negotiated Max Serv Outstanding Called			M	M(=)	
Negotiated Data Structure Nesting Level			U	U(=)	
Init Response Detail			M	M	
Result(-)			S	S(=)	
Error Type			M	M(=)	

NOTE – The collection of these parameters is contained in the user data portion of the A-ASSOCIATE request of ACSE. Additional parameters are required to complete the A-ASSOCIATE request. The mapping of the Initiate Service on to the A-ASSOCIATE is discussed in ISO/IEC 9506-2, clause 17. Owing to the mapping of Initiate on to A-Associate, resolution of collisions in the Initiate service is handled by ACSE mechanisms for resolving collisions in A-Associate.

8.2.1.1 Argument

This parameter shall convey the service specific parameters of the Initiate service request.

For those parameters in the request primitive, the MMS-provider in the calling system may reduce the values supplied by the calling MMS-user. For those parameters in the indication primitive, the MMS-provider in the called system may reduce the values from those in the InitiateRequestPDU, except for the Services Supported Calling parameter. Reductions of the values must follow the rules for reduction as specified in the parameter descriptions. No other modifications are permitted to these values.

NOTE – The purpose of allowing the service providers to reduce values as specified by the user is to provide a mechanism which may be used to ensure that the values specified by the user do not exceed the capabilities of the service provider. Implementation of this feature is a local matter.

8.2.1.1.1 Local Detail Calling

When present, this parameter, of type integer, shall represent information about the Calling MMS-User's implementation. The content and interpretation of this field is a local matter and not a subject for further standardization.

8.2.1.1.2 Proposed Max Serv Outstanding Calling

This parameter, of type integer, shall identify the proposed maximum number of Transaction Object instances whose Application Association Identifier attribute references this association that may be created at the Calling MMS-user.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.3 Proposed Max Serv Outstanding Called

This parameter, of type integer, shall identify the proposed maximum number of Transaction Object instances that may be created at the called MMS-user whose Application Association Identifier attribute references this association.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.4 Proposed Data Structure Nesting Level

This parameter, of type integer, shall indicate the proposed maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited number of nesting levels.

The request and indication primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that the calling MMS-user desires to use in the negotiated MMS environment. A value of zero shall indicate that only simple types are allowed (see 12.2).

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive.

8.2.1.1.5 Init Request Detail

This parameter shall specify additional information necessary for the establishment of an instance of MMS communication. For the abstract syntax defined in ISO/IEC 9506-1 and ISO/IEC 9506-2, this parameter is described in 8.2.3.

8.2.1.2 Result(+)

This parameter shall indicate that the requested service has succeeded. When success is indicated, the following additional parameters shall also apply.

8.2.1.2.1 Local Detail Called

When present, this parameter, of type integer, shall represent information about the Called MMS-User's implementation. The content of this field is a local matter and not a subject for further standardization.

8.2.1.2.2 Negotiated Max Serv Outstanding Calling

This parameter, of type integer, shall identify the maximum number of Transaction Object instances that may be created at the calling MMS-user whose Application Association Identifier attribute references this association.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Calling parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.3 Negotiated Max Serv Outstanding Called

This parameter, of type integer, shall identify the maximum number of Transaction Object instances that may be created at the called MMS-user whose Application Association Identifier attribute references this association.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Called parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.4 Negotiated Data Structure Nesting Level

This parameter, of type integer, shall indicate the maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited nesting level.

This parameter shall be equal to or less than the Proposed Data Structure Nesting level parameter in the indication primitive, but shall not be less than zero.

The response and confirmation primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that shall be used in the negotiated MMS environment. A value of zero shall indicate only simple types are allowed (see 12.2).

8.2.1.2.5 Init Response Detail

This parameter shall specify additional information necessary for the establishment of an instance of MMS communication. For the abstract syntax defined in ISO/IEC 9506-1 and ISO/IEC 9506-2 this parameter is described in 8.2.4.

8.2.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.2.2 Service Procedure

The called MMS-user shall issue a response primitive indicating success in the Result parameter if that MMS-user is willing to accept communications in the MMS environment under the constraints identified in the indication primitive, or if alternate values can be proposed (according to the negotiation rules), with the requesting MMS-user. Otherwise, a response primitive indicating the Result(-) parameter shall be issued.

Successful execution of the Initiate service shall result in the establishment of the MMS environment. The MMS environment shall only be established through the use of the Initiate service. The Initiate service shall not be used within an established MMS environment. If an Initiate Request PDU is received on an association where an established MMS environment exists, a Reject PDU shall be issued with a Reject PDU Type of PDU-ERROR and a Reject Code of ILLEGAL-ACSE-MAPPING.

8.2.3 Init Request Detail Parameter

The. structure of the Init Request Detail Parameter is shown in Table 4.

Table 4 – Init Request Detail Parameter

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Proposed Version Number	M	M			
Proposed Parameter CBB	M	M			
Services Supported Calling	M	M			

8.2.3.1 Proposed Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2. The minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be that specified in ISO/IEC 9506-2, clause 17. This parameter, the Proposed Version Number, is the proposed minor version number which will be used in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, for this instance of communication. Proposal of a number greater than one indicates support, in this presentation context, for all minor versions between one and the number proposed, inclusive.

NOTE – Major versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 are reflected through the definition and registration of distinct abstract syntaxes. (see ISO/IEC 9506-2, clauses 7 and 19). Minor versions are reflected in the minor version number parameter. Minor versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 at the same major version level are compatible with major versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 with smaller minor version numbers.

The value of this parameter may be reduced by the MMS-provider if it cannot support the requested value. The value in the indication primitive shall be less than or equal to the value in the request primitive, but not less than one.

8.2.3.2 Proposed Parameter CBB

This parameter, of type bitstring, shall specify the set of parameter conformance building blocks (CBB) which are proposed to be supported in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on this application association.

The value of this parameter in the request primitive shall specify the set of Parameter CBBs supported by the Calling MMS-user.

The value of this parameter in the indication primitive shall specify the intersection of the set of Parameter CBBs supported by the Calling MMS-user and the set of Parameter CBBs supported by the MMS-provider.

The possible parameter conformance building blocks are specified in ISO/IEC 9506-2, clause 18. The assignment of a parameter CBB to an individual bit of a CBB bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.3.3 Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of services supported by the Calling MMS-user and the set of services supported by the MMS-provider.

The assignment of an service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to execute properly the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

NOTE – This parameter only represents the services defined in this part of ISO/IEC 9506. Additional services defined in Companion Standards are represented in a separate parameter (see annex A).

8.2.4 Init Response Detail Parameter

The structure of the Init Response Detail Parameter is shown in Table 5.

Table 5 – Init Response Detail Parameter

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Negotiated Version Number			M	M(=)	
Negotiated Parameter CBB			M	M(=)	
Services Supported Called			M	M	

8.2.4.1 Negotiated Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2. The minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be that specified in ISO/IEC 9506-2, clause 17. This parameter, the Negotiated Version Number, shall be the minor version number which will be used in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, for this instance of communication. This number shall be less than or equal to the Proposed Version Number parameter in the request primitive. It shall not be reduced to less than one.

NOTE – Major versions of ISO/IEC 9506 are reflected through the definition and registration of distinct abstract syntaxes (see ISO/IEC 9506-2, clauses 7 and 19). Minor versions are reflected in the minor version number parameter. Minor versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 at the same major version level are compatible with versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 with smaller minor version numbers.

8.2.4.2 Negotiated Parameter CBB

This parameter, of type bitstring, shall specify the negotiated set of parameter conformance building blocks (CBB) which are to be supported in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on this application association.

The value of this parameter in the response primitive shall specify the intersection of the set of Parameter CBBs supported by the Called MMS-user and the set of Parameter CBBs specified by the Proposed Parameter CBB parameter in the indication primitive.

The possible parameter conformance building blocks are specified in ISO/IEC 9506-2, clause 18. The assignment of a Parameter CBB to an individual bit of a CBB bitstring type is specified in of ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.4.3 Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on the application association.

The value of the parameter in the confirmation primitive shall specify the intersection of the set of services supported by the Called MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive an indication primitive and properly execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to properly execute the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

NOTE – This parameter only represents the services defined in this part of ISO/IEC 9506. Additional services defined in Companion Standards are represented in a separate parameter (see annex A).

8.3 Conclude Service

The Conclude service may be used to cause the orderly relinquishing of the MMS environment. An MMS-user requests the Conclude service to indicate that it has completed requests which it had planned and that no further requests will be issued.

8.3.1 Structure

The structure of the component service primitives is shown in Table 6.

Table 6 – Conclude Service

Table 6 (Cont.) – Conclude Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

8.3.1.1 Argument

There are no service specific parameters for the Conclude service request.

8.3.1.2 Result(+)

This parameter shall indicate that the requested service has succeeded.

8.3.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.3.2 Service Procedure

The requesting MMS-user shall issue a Conclude request primitive, which begins the Conclude service. Once this primitive has been issued, the requesting MMS-user shall not issue any further request primitives until a Conclude confirm primitive is received from the MMS-provider (except the Abort request primitive, which may be issued at any time). The requesting MMS-user may continue to issue response primitives in order to complete service requests from the peer MMS-user.

The requesting MMS-user shall not issue a Conclude request primitive to a peer on an association if a Domain with a State attribute value equal to LOADING, COMPLETE, INCOMPLETE, D1, D2, D3, or D9 exists at the requesting MMS-user as a result of a request from the peer MMS-user on the association.

In the peer open system, the responding MMS-user shall issue a Conclude response primitive indicating whether or not it accepts the conclusion (via the Result parameter) before any other service request primitives may be issued by that MMS-user (except the Abort request service primitive, which may be issued at any time). The MMS-user shall not accept conclusion on an association if it is awaiting any response from the peer MMS-user for a confirmed service or the Conclude Service.

A responding MMS-user shall not accept a Conclude attempt on an association if any of the following conditions exist:

a) The responding MMS-user has received an indication on the association for a confirmed service request from the peer MMS-user for which it has not issued a response;

b) An Upload State Machine exists on the responding MMS-user as a result of a request from the peer MMS-user on the association;

c) A Domain with a State attribute value equal to LOADING, COMPLETE, or INCOMPLETE exists on the responding MMS-user as a result of a request from the peer MMS-user on the association;

d) The responding MMS-user has control of a semaphore on the requesting MMS-user on the association;

e) The requesting MMS-user has control of a semaphore on the responding MMS-user on the association.

Upon issuing a Conclude response primitive with a Result(+) parameter, the MMS Environment shall be considered terminated. No further request or response primitives shall be issued by the responding MMS-user in that MMS environment on that association.

Upon issuing a Conclude response primitive with a Result(-), the MMS environment shall not be affected and the responding MMS-user may continue to issue request and/or response service primitives.

Upon receiving a Conclude confirm primitive with a Result(+), the requesting MMS-user shall consider the MMS environment to be terminated, and no further request or response primitives shall be issued by the requesting MMS-user in the MMS environment on that association.

Upon receiving a Conclude confirmation primitive with a Result(-), the requesting MMS-user shall consider the MMS Environment to be unaffected, and may continue to issue request or response service primitives in the MMS environment.

NOTE 1 – The effect of a failed Conclude service confirmation is as if no Conclude request had ever been issued.

Upon successful completion of the Conclude service, all application association specific objects associated with the terminated MMS environment are released unless other specifications for that object are explicitly stated in this part of ISO/IEC 9506.

NOTE 2 – It is possible that both of the peers will initiate a Conclude attempt at or near the same time resulting in failure of both attempts. If both peers retry the Conclude, on result of failure, then this situation could potentially (although very unlikely) continue indefinitely. Through use of application design, this situation can be avoided. One possible method is to designate the Calling MMS-user as the peer that will initiate the Conclude attempt should this situation arise.

8.4 Abort Service

The Abort service shall be used to relinquish the MMS environment abruptly and without negotiation. The MMS-user shall issue the Abort request primitive to indicate that it wishes immediately, and without negotiation, to discontinue communications in the MMS Environment on the application association. The effect of the Abort service may be to destroy previously issued requests and/or responses issued by either of the MMS-users. The MMS Abort service is derived from and makes use of the A-ABORT service of ACSE (see 17.2 in ISO/IEC 9506-2).

NOTE – The abort indication service primitive may also be generated by the MMS-provider.

8.4.1 Structure

The structure of the component service primitives is shown in Table 7.

Table 7 – Abort Service

Parameter Name	Req	Ind	CBB
Argument	M	M	
Locally Generated		M	

8.4.1.1 Argument

This parameter shall convey the service specific parameters of the Abort service. The abort service primitives additionally shall contain the parameters specified for the ACSE abort service to which the MMS Abort is mapped. These parameters are specified in ISO 8649.

8.4.1.1.1 Locally Generated

This parameter, of type boolean, shall indicate whether the abort request was generated by the system in which the MMS-user receiving the indication is located (indicated by the true value), or whether the abort request was received by that system (indicated by the false value). This parameter shall be provided by the MMS-provider.

8.4.2 Service Procedure

In the case of an Abort initiated by the MMS-user, the peer MMS-user shall be notified of the user requested abort by receipt of an Abort indication service primitive, and the MMS Environment shall be terminated.

In the case of an Abort initiated by the MMS-provider, both MMS-users shall be notified of the provider abort by receipt of Abort indication primitives (if this is possible), and the MMS Environment shall be terminated.

Upon completion of the Abort service, all application association specific objects associated with the terminated MMS environment are released unless other specifications for that object are explicitly stated in ISO/IEC 9506.

8.5 Cancel Service

The Cancel service is used by an MMS-user to cancel a request that has previously been issued, but has not yet completed. Only confirmed services may be cancelled (see clause 5).

8.5.1 Structure

The structure of the component service primitives is shown in Table 8.

Table 8 – Cancel Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Original Invoke ID	M	M(=)			
Result(+)			S	S(=)	
Original Invoke ID			M	M(=)	
Result(-)			S	S(=)	
Original Invoke ID			M	M(=)	
Error Type			M	M(=)	

8.5.1.1 Argument

This parameter shall convey the service specific parameters for the Cancel service request.

8.5.1.1.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation is desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.2 Result(+)

The Result(+) parameter shall indicate that the Cancel service request succeeded. When success is indicated service specific parameters shall also be returned.

8.5.1.2.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service that was cancelled. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.3 Result(-)

The Result(-) parameter shall indicate that the Cancel service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.5.1.3.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation was desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.3.2 Error Type

The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

8.5.2 Service Procedure

Upon receiving the Cancel indication service primitive, the responding MMS-user shall attempt to cancel the designated service, assuming that it can do this non-destructively and can return to a state that is logically as if the service indication targeted for cancellation had not been received. Destructive cancellation is permitted for the Start, Stop, Resume, and Reset Services provided it follows the service procedures defined for these services. If this is not the case, or if the service request invocation identified has not been requested, or if a response primitive has already been issued, or if the cancelable attribute of the Transaction object associated with the designated service has a value of false, then the Cancel request shall fail. In this case, the responding MMS-user shall issue a Cancel response service primitive with the Result(-) parameter with the appropriate Cancel error codes for the Error Type parameter. The responding MMS-user shall continue with the original service and eventually return a response, either positive or negative, as appropriate.

Otherwise, the service invocation identified shall be successfully cancelled by the responding MMS-user, and a Cancel response primitive with the Result(+) parameter shall be issued. The responding MMS-user shall also issue a response primitive with a Result(-) parameter for the cancelled service indicating the SERVICE-PREEMPT Error Class with an Error Code of CANCEL. The state of the responder shall then appear logically as if the service targetted for cancellation had not been requested, unless otherwise stated in the service procedure for the cancelled service. Additional limitations on requesting the cancel service are specified in ISO/IEC 9506-2, clause 6.

8.6 Reject Service

The Reject service is a provider-initiated service that is used to inform the MMS-users of the occurrence of a protocol error. A definition of a protocol error may be found in ISO/IEC 9506-2, clause 3.

8.6.1 Structure

The structure of the component service primitives is shown in Table 9.

Table 9 – Reject Service

Parameter Name	Ind	CBB
Detected Here	M	
Original Invoke ID	C	
Reject PDU Type	M	
Reject Code	M	

8.6.1.1 Detected Here

This parameter, of type boolean, shall indicate whether the protocol error that results in the Reject service was detected at the local end or remote end of the underlying presentation connection. If true, the protocol error was detected at the local end, otherwise the error was detected at the remote end.

NOTE – Handling of local error conditions between an MMS-user and MMS-provider is a local matter.

8.6.1.2 Original Invoke ID

This parameter, of type integer, shall indicate the original invoke ID of the PDU which was found to be in error. Its presence is conditional on whether the invoke ID could be determined. If there is no invoke ID specified in the PDU being rejected, this parameter shall not be present.

8.6.1.3 Reject PDU Type

This parameter, of type integer, shall indicate the type of the PDU that caused the protocol error. The value PDU-ERROR shall be used when the PDU being rejected is not a syntactically valid MMS PDU. The possible values are as follows:

CONFIRMED-REQUESTPDU
CONFIRMED-RESPONSEPDU
CONFIRMED-ERRORPDU
UNCONFIRMEDPDU
PDU-ERROR
CANCEL-REQUESTPDU
CANCEL-RESPONSEPDU
CANCEL-ERRORPDU
CONCLUDE-REQUESTPDU
CONCLUDE-RESPONSEPDU
CONCLUDE-ERRORPDU

8.6.1.4 Reject Code

This parameter, of type integer, shall indicate additional information regarding the reason for the Reject within a given Reject PDU Type parameter value. The possible values for this parameter are given below under each possible Reject PDU Type parameter value.

8.6.1.4.1 Codes for CONFIRMED-REQUESTPDU

8.6.1.4.1.1 OTHER

This code shall be used for errors other than those identified in this part of ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.1.2 UNRECOGNIZED-SERVICE

This code shall be used when the service requested is not supported or is not recognized.

8.6.1.4.1.3 UNRECOGNIZED-MODIFIER

This code shall be used when a modifier requested is not supported or is not recognized.

8.6.1.4.1.4 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506.

8.6.1.4.1.5 INVALID-ARGUMENT

This code shall be used when a service argument does not meet the requirements of ISO/IEC 9506.

8.6.1.4.1.6 INVALID-MODIFIER

This code shall be used when a service modifier does not meet the requirements of ISO/IEC 9506.

8.6.1.4.1.7 MAX-SERV-OUTSTANDING-EXCEEDED

This code shall be used when the negotiated maximum number of confirmed services that may be outstanding is exceeded by a confirmed service request that is received.

8.6.1.4.1.8 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.1.9 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506 .

8.6.1.4.2 Codes for CONFIRMED-RESPONSEPDU

8.6.1.4.2.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.2.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported, is not recognized, or is not the same service that was requested with the invoke ID specified in the PDU.

8.6.1.4.2.3 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506, or no confirmed service has been requested for the specified invoke ID.

8.6.1.4.2.4 INVALID-RESULT

This code shall be used when a service result does not meet the requirements of ISO/IEC 9506.

8.6.1.4.2.5 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.2.6 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506.

8.6.1.4.3 Codes for CONFIRMED-ERRORPDU

8.6.1.4.3.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.3.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported, is not recognized, or is not the same service that was requested with the invoke ID specified in the PDU.

8.6.1.4.3.3 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506, or no confirmed service has been requested for the specified invoke ID.

8.6.1.4.3.4 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of ISO/IEC 9506.

8.6.1.4.3.5 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506.

8.6.1.4.4 Codes for UNCONFIRMEDPDU

8.6.1.4.4.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.4.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported or is not recognized.

8.6.1.4.4.3 INVALID-ARGUMENT

This code shall be used when a service argument does not meet the requirements of ISO/IEC 9506.

8.6.1.4.4.4 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.4.5 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506.

8.6.1.4.5 Codes for PDU-ERROR

8.6.1.4.5.1 UNKNOWN-PDU-TYPE

This code shall be used when the PDU type received is not recognized or is not supported.

8.6.1.4.5.2 INVALID-PDU

This code shall be used when the PDU received is syntactically incorrect, and further diagnostics cannot be provided due to the severity of the error.

8.6.1.4.5.3 ILLEGAL-ACSE-MAPPING

This code shall be used when the PDU type received is not properly mapped to an ACSE service primitive.

8.6.1.4.6 Codes for CANCEL-REQUESTPDU

8.6.1.4.6.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.6.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506.

8.6.1.4.7 Codes for CANCEL-RESPONSEPDU

8.6.1.4.7.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.7.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506, or no Cancel service has been requested for the specified invoke ID.

8.6.1.4.8 Codes for CANCEL-ERRORPDU

8.6.1.4.8.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.8.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of ISO/IEC 9506, or no Cancel service has been requested for the specified invoke ID.

8.6.1.4.8.3 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of ISO/IEC 9506.

8.6.1.4.8.4 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506.

8.6.1.4.9 Codes for CONCLUDE-REQUESTPDU

8.6.1.4.9.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.9.2 INVALID-ARGUMENT

This code shall be used when the argument for the request does not meet the requirements of ISO/IEC 9506.

8.6.1.4.10 Codes for CONCLUDE-RESPONSEPDU

8.6.1.4.10.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.10.2 INVALID-RESULT

This code shall be used when the service result does not meet the requirements of ISO/IEC 9506.

8.6.1.4.11 Codes for CONCLUDE-ERRORPDU

8.6.1.4.11.1 OTHER

This code shall be used for errors other than those identified in ISO/IEC 9506 for this Reject PDU type.

8.6.1.4.11.2 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of ISO/IEC 9506.

8.6.1.4.11.3 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by ISO/IEC 9506.

8.6.2 Service Procedure

If an MMS-provider receives a RejectPDU it shall issue a Reject indication to the MMS-user. The MMS-user may, as a local matter, use the Abort service to abruptly terminate the MMS environment.

An MMS provider shall be capable of issuing a RejectPDU if it receives a PDU that constitutes a protocol error.

9 VMD Support Services

9.1 Introduction

The VMD support services contain the Status, UnsolicitedStatus, GetNameList, Identify, Rename and GetCapabilityList services. The services allow the MMS-user to do the following:

a) get the status of a VMD;

b) receive an unsolicited message about the status of the VMD;

c) get lists of various defined objects;

d) identify the vendor specific attributes of the MMS application entity at the peer system;

e) change the name of an object;

f) get lists of the VMD's capabilities.

9.2 Status Service

The purpose of the Status service is to allow an MMS-user to determine the general condition or status of a VMD.

9.2.1 Structure

The structure of the component service primitives is shown in Table 10.

Table 10 – Status Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Extended Derivation		M	M(=)			
Result(+)	(COMP)			S	S(=)	
VMD Logical Status				M	M(=)	
VMD Physical Status				M	M(=)	
Local Detail				U	U(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

9.2.1.1 Argument

This parameter shall convey the parameter of the Status service request.

9.2.1.1.1 Extended Derivation

This parameter, of type boolean, shall indicate which method is used to derive the Status response. This parameter is applicable only if the VMD supports two methods for deriving the Status Response, where one method results in a more extensive derivation of the Status Response. If the value of this parameter is true, then the method that results in a extended derivation shall be used. If the value is false, then the other method shall be used. If the VMD only supports one method, then the value shall be ignored.

NOTE – An example of an extended derivation method is the invocation of a set of self-diagnostic routines.

9.2.1.2 Result(+)

The Result(+) parameter shall indicate that the Status service request succeeded. When success is indicated, service specific parameters are also returned.

9.2.1.2.1 VMD Logical Status

This parameter, of type integer, shall convey the value of the Logical Status attribute of the VMD object. The VMD object is defined in 7.2.

9.2.1.2.2 VMD Physical Status

This parameter, of type integer, shall convey the value of the Physical Status attribute of the VMD object. The VMD object is defined in 7.2.

9.2.1.2.3 Local Detail

When present, this parameter, of type bitstring, shall represent additional detail about the status of the VMD as specified by the vendor of the device. This parameter is expressed as a bitstring no greater than 128 bits in length. The content of this field is a local matter and is not a subject for further standardization.

9.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

9.2.2 Service Procedure

The responding MMS-user shall perform the Status service by determining the information necessary to create a valid response.

9.3 UnsolicitedStatus Service

The UnsolicitedStatus service may be used by an MMS-user to spontaneously report its status.

9.3.1 Structure

The structure of the component service primitives is shown in Table 11.

Table 11 – UnsolicitedStatus Service

Parameter Name		Req	Ind	CBB
Argument	(COMP)	M	M(=)	
VMD Logical Status		M	M(=)	
VMD Physical Status		M	M(=)	
Local Detail		U	U(=)	

The parameters in this service have equivalent meaning to the parameters in the Status response and confirm service primitives.

9.3.2 Service Procedure

An MMS-user which is capable of detecting a change in its own status may, at its option, report this change without receiving a Status request. This is accomplished by requesting the UnsolicitedStatus service. The information in the UnsolicitedStatus.request shall reflect the values of the corresponding attributes of the VMD. A UnsolicitedStatus.request shall not be sent if the peer MMS-user did not indicate support of the UnsolicitedStatus service in the Services Supported parameter received in the Initiate Service.

NOTE – The choice of application associations (if more than one exists) on which to send the UnsolicitedStatus Service request is a local matter (which may be further specified by Companion Standards). All associations, one association, or some group may be selected.

The use of this service is functionally equivalent to an Event Notification with an Event Action of the Status Service. The practical difference is that by using the Event Notification method, the conditions under which the service is used are directly visible (and modifiable) using MMS services, while in use of the UnsolicitedStatus, the conditions are a local matter and cannot be determined or changed by the MMS user that is to receive the UnsolicitedStatus information.

9.4 GetNameList Service

The GetNameList service may be used by an MMS-user to request that a responding MMS-user return the list of or part of the list of object names defined at the VMD.

9.4.1 Structure

The structure of the component service primitives is shown in Table 12.

Table 12 – GetNameList Service

Table 12 (Cont.) – GetNameList Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Extended Object Class	M	M(=)			
Object Class	S	S(=)			
CS Object Class	COMP	COMP			
Object Scope	M	M(=)			
Domain Name	C	C(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List Of Identifier			M	M(=)	
More Follows			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

9.4.1.1 Argument

This parameter shall convey the parameters of the GetNameList service request.

9.4.1.1.1 Extended Object Class

This parameter shall specify the extended object class of the object name to be returned by the responding MMS-user. The possible values shall be OBJECT-CLASS, which specifies a choice among objects specified in this part of ISO/IEC 9506, or CS-OBJECT-CLASS, which specifies a choice among objects specified in a Companion Standard.

9.4.1.1.1.1 Object Class

This parameter shall specify the class of the object name to be returned by the responding MMS-user. It specifies a choice of the following objects:

Named Variable

Scattered Access

Named Variable List

Named Type

Semaphore

Event Condition

Event Action

Event Enrollment

Journal

Domain

Program Invocation

Operator Station

9.4.1.1.1.2 CS Object Class

The CS Object Class parameter is provided for definition by a Companion Standard and shall specify the class of the object name to be returned by the responding MMS-user. The values for object name shall be limited to any additional objects defined in a Companion Standard and may be further defined by that standard. This choice of the Extended Object Class shall not be chosen in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19.

NOTE – Further information may be found in annex A.

9.4.1.1.2 Object Scope

This parameter shall indicate the scope of the object name list to be returned. The possible values are VMD-specific, Domain-specific, and AA-specific.

9.4.1.1.3 Domain Name

This parameter, of type Identifier, shall specify the name of a Domain if the Domain-specific choice of the Object Scope parameter is chosen. Otherwise, this parameter shall not be present.

9.4.1.1.4 Continue After

This parameter, of type identifier, shall be present when the MMS-user wishes the list of object names returned by the responding MMS-user to begin with a name other than the (logical) first name in the list. It shall be a character string containing the name of an object of class and scope specified in the Extended Object Class, Object Class and Object Scope parameters. If the value of the Continue After parameter does not match an existing name at the VMD of the class and scope specified, then the collating sequence of the International Reference Version of ISO 646 shall be used by the responding MMS-user to determine the name to start after as specified in the service procedure.

9.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated, service specific parameters shall also be returned.

9.4.1.2.1 List Of Identifier

This parameter, of type identifier, shall contain names that exist for the objects of the class and scope specified in the Object Class and Object Scope parameters subject to reduction by the action of the Continue After parameter. The returned list shall contain zero or more entries and shall be sorted according to the collating sequence of the International Reference Version of ISO 646.

9.4.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional GetNameList requests are necessary to retrieve all of the requested information. If true, more requests are necessary (if the requesting MMS-user wishes to retrieve more data). If false, then either the List Of Names contains the last name in the list, or the List Of Identifier is empty. This parameter shall be false when the List Of Identifier parameter contains zero names.

9.4.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

9.4.2 Service Procedure

The responding MMS-user shall return the List Of Identifier of the class and scope specified in the Extended Object Class and Object Scope parameters of the indication service primitive. If the Continue After parameter is not present in the indication service primitive, List Of Identifier shall begin with the first name, as determined by the collating sequence of the International Reference Version of ISO 646, in the List Of Objects at the responding MMS-user satisfying the class and scope specifications. Otherwise, the List Of Identifier shall begin with the first name after the name specified in the Continue After parameter, using the ordering described above. The More Follows parameter shall be returned with the value determined as described above.

The responding MMS-user shall not issue a Result(-) parameter if no objects of the requested class and scope exist. Instead, a Result(+) parameter shall be issued with an empty list of Identifiers.

NOTE – Repeated usage of this service with the Continue After parameter does not guarantee a list that is consistent in time with any other instance of use of this service. This service returns a segment of the list of names for which the starting point is based on the Continue After parameter.

9.5 Identify Service

The Identify service may be used by an MMS-user to obtain identifying information from a responding MMS-user.

9.5.1 Structure

The structure of the component service primitives is shown in Table 13.

Table 13 – Identify Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Vendor Name			M	M(=)	
Model Name			M	M(=)	
Revision			M	M(=)	
List Of Abstract Syntaxes			C	C(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

9.5.1.1 Argument

There are no service specific parameters of the Identify service request.

9.5.1.2 Result(+)

The Result(+) parameter shall indicate that the Identify service request succeeded. When success is indicated, service specific parameters shall also be returned.

9.5.1.2.1 Vendor Name

This parameter, of type character string, shall convey the value of the vendor name attribute of the VMD object. The VMD object is defined in 7.2.

9.5.1.2.2 Model Name

This parameter, of type character string, shall convey the value of the model name attribute of the VMD object. The VMD object is defined in 7.2.

9.5.1.2.3 Revision

This parameter, of type character string, shall convey the value of the revision attribute of the VMD object. The VMD object is defined in 7.2.

9.5.1.2.4 List Of Abstract Syntaxes

When included, this parameter, of type object identifier, shall convey the value of the List Of Abstract Syntaxes Supported attribute of the VMD object. The VMD object is defined in 7.2. This parameter shall not be included if the list contains zero abstract syntax names.

9.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

9.5.2 Service Procedure

A response service primitive with the Result(+) parameter shall be issued providing the specified information.

9.6 Rename Service

The Rename service may be used by an MMS-user in order to request that a responding MMS-user change the identifier of an object to a specified identifier.

9.6.1 Structure

The structure of the component service primitives is shown in Table 14.

Table 14 – Rename Service

Table 14 (Cont.) – Rename Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Extended Object Class	M	M(=)			
Object Class	S	S(=)			
CS Object Class	COMP	COMP			
Current Name	M	M(=)			
New Identifier	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

9.6.1.1 Argument

This parameter shall convey the service specific parameters of the Rename service request.

9.6.1.1.1 Extended Object Class

This parameter shall specify the extended object class of the object name to be renamed by the responding MMS-user. The possible values shall be OBJECT-CLASS, which specifies a choice among objects specified in this part of ISO/IEC 9506, or CS-OBJECT-CLASS, which specifies a choice among objects specified in a Companion Standard.

9.6.1.1.1.1 Object Class

This parameter shall specify the class of the object to be renamed. It shall specify a choice of the following objects:

Named Variable

Scattered Access

Named Variable List

Named Type

Semaphore

Event Condition

Event Action

Event Enrollment

Journal

Domain

Program Invocation

Operator Station

9.6.1.1.1.2 CS Object Class

The CS Object Class parameter is provided for definition by a Companion Standard and shall specify the class of the object name to be returned by the responding MMS-user. The values for object name shall be limited to any additional objects defined in a Companion Standard and may be further defined by that standard. This choice of the Extended Object Class shall not be chosen in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19.

NOTE – Further information may be found in annex A.

9.6.1.1.2 Current Name

This parameter, of type object name, shall specify the object name of the object that is to be renamed.

9.6.1.1.3 New Identifier

This parameter, of type identifier, shall specify the new identifier part for the referenced object name.

9.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded.

9.6.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

9.6.2 Service Procedure

The responding MMS-user shall verify that an object exists with the type specified in Extended Object Class parameter and the object name specified in the Current Name parameter. It then shall verify that an object does not exist with the same class type and scope of the Current Name parameter and identifier of the New Identifier parameter. If the new identifier identifies an object of the classes "Named Variable" or "Scattered Access", it shall verify that an object of these classes does not exist with the same scope of the Current Name parameter and identifier of the New Identifier parameter. If and only if both of these criteria are satisfied, it shall change the identifier of the specified object to that supplied in the New Identifier parameter. No service specific parameters shall be returned.

NOTE – The Rename service is provided to augment commissioning facilities to change the name of objects that are supplied as part of a VMD. Indiscriminate use of the service could cause existing applications to fail if the identifiers of referenced objects are renamed.

9.7 GetCapabilityList Service

The GetCapabilityList service may be used by an MMS-user in order to request that a responding MMS-user return the list of or part of the list of capabilities defined at the VMD.

9.7.1 Structure

The structure of the component service primitives is shown in Table 15.

Table 15 – GetCapabilityList Service

Table 15 (Cont.) – GetCapabilityList Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Continue After		U	U(=)			
Result(+)	(COMP)			S	S(=)	
List Of Capabilities				M	M(=)	
More Follows				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

9.7.1.1 Argument

This parameter shall convey the parameters of the GetCapabilityList service request.

9.7.1.1.1 Continue After

This parameter, of type character string, shall be present when the MMS-user wishes the List Of Capabilities returned by the responding MMS-user to begin with a capability other than the first capability in the list. If the Continue After parameter does not match an existing capability of the VMD, then the collating sequence of the International Reference Version of ISO 646 shall be used by the responding MMS-user to determine the capability to start after.

9.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated, service specific parameters shall also be returned.

9.7.1.2.1 List Of Capabilities

This parameter, of type character string, shall convey the value of the List of Capabilities attribute of the VMD object. The VMD object is defined in clause 7.2. The returned list shall contain zero or more entries and shall be sorted according to the collating sequence of the International Reference version of ISO 646.

9.7.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional GetCapabilityList requests are necessary to retrieve more of the requested information. If true, more requests are necessary (if the requesting MMS-user wishes to retrieve more data). If false, then either the List Of Capabilities contains the last capability in the list, or the List Of Capabilities is empty. The More Follows parameter shall be false when the List Of Capability parameter contains zero capabilities.

9.7.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

9.7.2 Service Procedure

The responding MMS-user shall return the List Of Capabilities attribute of the VMD. If the Continue After parameter is not present in the indication service primitive, the List Of Capabilities parameter shall begin with the first capability, as determined by the collating sequence of the International Reference Version of ISO 646, in the VMD attribute List Of Capabilities at the responding MMS-user. Otherwise, the List Of Capabilities parameter shall begin with the first capability after the capability specified in the Continue After parameter , using the ordering described above. The More Follows parameter shall be returned with the value determined as described above.

10 Domain Management Services

The MMS model of the Virtual Manufacturing Device (VMD) described in clause 7 introduces four abstract elements: Executive Function, Capabilities, Program Invocations, and Domains. This clause describes the services which manage Domains. Domains may be dynamic in nature, coming into existence and being removed from the system either by MMS services or by local action. Services are provided to allow an MMS client to manipulate Domains defined at the MMS server.

10.1 The Domain Object

A Domain represents a subset of the capabilities of the VMD which is used for a specific purpose. The attributes of the Domain are described below, followed by a brief description of the services which operate on the Domain object.

Domains come into existence in one of two distinct ways: (1) A Domain is explicitly created as part of the process of beginning a download process. Such a Domain is inherently associated with its content. (2) A Domain may be created as part of the execution of a Program Invocation or by other local means. Domains may also be predefined to the system, existing prior to the establishment of an MMS context.

10.1.1 The Domain Object Model

Object: Domain

```
Key Attribute: Domain Name
Attribute: List Of Capabilities
Attribute: State (LOADING,COMPLETE,INCOMPLETE,
           READY,IN-USE,D1,D2,D3,D4,D5,D6,D7,D8,D9)
Constraint: State = (LOADING,COMPLETE,INCOMPLETE,D1,D2,D3,D9)
     Attribute: Assigned Application Association
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Sharable (TRUE, FALSE)
Attribute: Domain Content
     Attribute: List Of Subordinate Objects
Attribute: List Of Program Invocation References
Constraint: State = (IN-USE,READY,D4,D5,D6,D7,D8)
     Attribute: Upload In Progress
Attribute: Additional Detail
```

Domain Name

The Domain Name attribute uniquely identifies the Domain within the VMD. The Domain Name shall be a VMD-specific Object Name formed according to the rules for MMS Object Names as specified in 7.4.

List Of Capabilities

The List Of Capabilities attribute is a list of implementation specific parameters necessary to partition the total resources of the VMD for a Domain. The value of elements of this list, represented as character strings, are a local matter.

NOTE – The intent of Capability is to convey such parameters as memory allocation, processor assignments, and Input Output bindings.

State

The State attribute specifies the state of the Domain. Each Domain may be in one of five states: LOADING, COMPLETE, INCOMPLETE, READY, or IN-USE. The possible values of the State attribute depend on the method of creating the Domain. Prior to its creation, the Domain is non-existent. In order to complete the state table, a non-existent Domain is described as being in the NON-EXISTENT state. The LOADING state is an intermediate state which occurs during the loading process. The Domain enters the READY state following a successful Download. The IN-USE state differs from the READY state in that one or more Program Invocations have been defined using this Domain. The COMPLETE state is an intermediate state which occurs after the last DownloadSegment has been received but before the Download Sequence has been terminated. The INCOMPLETE state is an intermediate state which occurs when a Download Sequence is terminated before the loading process is complete. States D1-D9 represent intermediate states, i.e. states between a request and its response.

Assigned Application Association

During the process of Downloading a Domain, the Domain is dependent on the Application Association over which the Domain was created. If the Application Association is lost before the Domain is placed in the READY state, the Domain shall be automatically deleted. There are no MMS services which report the value of this attribute.

MMS Deletable

The MMS Deletable attribute specifies whether (true) or not (false) this Domain Object may be deleted using the Delete-Domain service.

Sharable

The Sharable attribute specifies whether this Domain may be used in more than one Program Invocation definition at the same time.

NOTE – A Domain which is read-only, that is, which is not altered by the action of the Program Invocation, is normally sharable. In many cases, Domains which can be modified by execution of a Program Invocation are not sharable; however, by careful coordination a Domain could be modified by two Program Invocations simultaneously, thereby providing a means of inter-process communication. Sharable does not necessarily imply read-only.

Domain Content

The Domain Content attribute is the information contained in the load data which is the subject of the DownloadSegment and the LoadDomainContent services. The Domain Content includes the List Of Subordinate Objects described below. The nature of the information in the Domain Content is a local matter.

List Of Subordinate Objects

The List Of Subordinate Objects attribute is a list of references to named MMS objects which are part of the Domain Content. These objects come into existence with the creation of the Domain and are deleted when the Domain is deleted. This list includes all currently defined MMS objects which have Domain-specific scope naming this Domain (see 7.4).

List Of Program Invocation References

This attribute is a list of references to Program Invocations which currently use this Domain. If the Domain is not sharable, this list has at most one Program Invocation. If the Domain is in the IN-USE state, this list shall not be empty.

Upload In Progress

The Upload In Progress attribute specifies the number of Upload Sequences currently active for this Domain. This attribute shall be the number of ULSMs which exist for this Domain (see 10.1.4.2). The value zero indicates that no Upload is currently in progress for this Domain.

Additional Detail

This attribute contains zero or more attributes whose meaning and type are defined by appropriate Companion Standards.

10.1.2 Operations on Domains

The following services operate on Domain Objects:

InitiateDownloadSequence allows an MMS client to begin a download sequence to load a Domain at an MMS server. This service is described in 10.2.

DownloadSegment allows an MMS server to obtain elements of the download information. This service is described in 10.3.

TerminateDownloadSequence allows an MMS server to terminate the download sequence. This service is described in 10.4.

InitiateUploadSequence allows an MMS client to begin the process of uploading the content of a specific Domain. This service is described in 10.5.

UploadSegment allows an MMS client to obtain a segment of the upload from the MMS server. This service is described in 10.6.

TerminateUploadSequence allows an MMS client to terminate the upload sequence. This service is described in 10.7.

RequestDomainDownload is used by the MMS server to request that the MMS client initiate a Download Sequence for the server. This service is described in 10.8.

RequestDomainUpload is used by the MMS server to request that the MMS client initiate an Upload Sequence for the MMS server specifying the contents of a named Domain. This service is described in 10.9.

LoadDomainContent allows an MMS client to request that the MMS server take action to load a Domain. This load data may come from the server's own filestore, or it may cause the server to obtain the load data from a third party file server. This service is described in 10.10.

StoreDomainContent allows an MMS client to request that the MMS server take action to transfer a Domain content to a filestore (perform an upload). This service is described in 10.11.

DeleteDomain allows an MMS client to request that the MMS server delete the specified Domain and release associated resources. This service is described in 10.12.

GetDomainAttributes allows an MMS client to request the MMS server to provide a list of the attributes of the specified Domain. This service is described in 10.13.

10.1.3 Domain State Diagrams

In Figure 4, intermediate states (states which exist only between an indication primitive and the response primitive or between a request primitive and a confirm primitive) are indicated by boxes labeled Dn. While these states are transitory, the Domain may be in such a state for some period of time, and may be reported in the same manner as major states.

Domains which are predefined or which come into existence through local means are restricted to the READY and IN-USE states of this diagram. The term Program Invocation Count refers to the number of Program Invocations which are currently bound to this Domain. (See clause 11 for information regarding the binding of Program Invocations to Domains.)

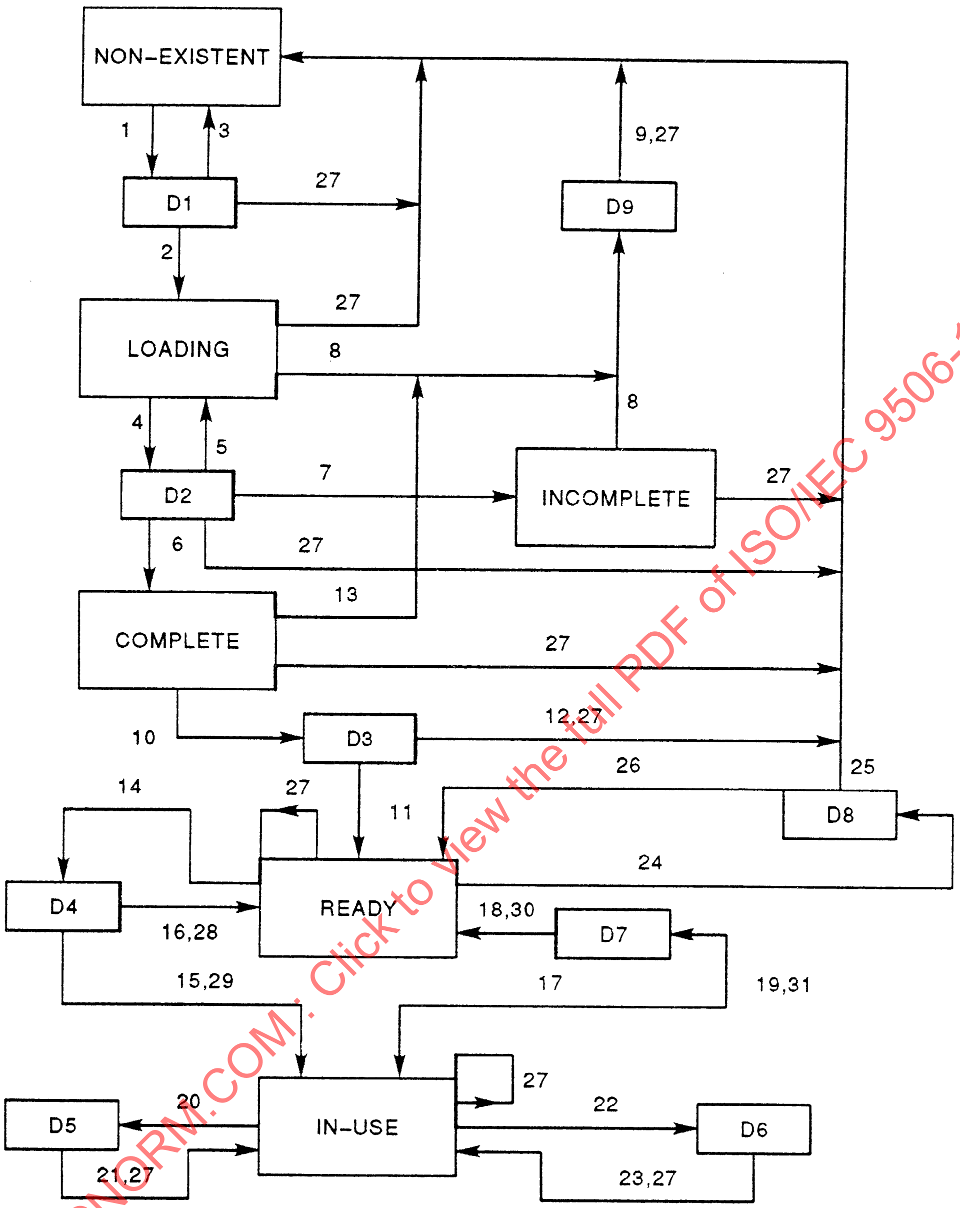


Figure 4 – Domain State Diagram

Transitions of the State Diagram are as follows:

1 - InitiateDownloadSequence.indication
2 - InitiateDownloadSequence.response (+)
3 - InitiateDownloadSequence.response (-)
4 - DownloadSegment.request
5 - DownloadSegment.confirm (+) More Follows = true
6 - DownloadSegment.confirm (+) More Follows = false
7 - DownloadSegment.confirm (-)
8 - TerminateDownloadSequence.request Discard present
9 - TerminateDownloadSequence.confirm (+) or (-)

10 - TerminateDownloadSequence.request Discard not present
11 - TerminateDownloadSequence.confirm (+)
12 - TerminateDownloadSequence.confirm (-)
13 - TerminateDownloadSequence.request Discard present
14 - CreateProgramInvocation.indication
Program Invocation count = 0
15 - CreateProgramInvocation.response (+)
16 - CreateProgramInvocation.response (-)
17 - DeleteProgramInvocation.indication
Program Invocation count = 1
18 - DeleteProgramInvocation.response (+)
19 - DeleteProgramInvocation.response (-)
20 - CreateProgramInvocation.indication
Program Invocation count > 0
21 - CreateProgramInvocation.response (+) or (-)
22 - DeleteProgramInvocation.indication
Program Invocation count > 1
23 - DeleteProgramInvocation.response (+) or (-)
24 - DeleteDomain.indication
25 - DeleteDomain.response (+)
26 - DeleteDomain.response (-)
27 - Abort.indication
28 - Abort.indication Program Invocation creation failed
29 - Abort.indication Program Invocation creation succeeded
30 - Abort.indication Program Invocation deletion succeeded
31 - Abort.indication Program Invocation deletion failed

10.1.4 Segmented Services

There are two sets of services within Domain Management in which the services are required to occur in groups. These are the Download Sequence services and the Upload Sequence services.

10.1.4.1 Download Sequence

The Domain Download Sequence may be used to accomplish the creation and loading of Domain Content from the MMS client to the MMS server. Although the MMS client initiates this sequence by requesting the InitiateDownloadSequence Service, subsequent services are controlled by the MMS server. The MMS server shall issue zero or more DownloadSegment Service requests (as required) followed by a request for the TerminateDownloadSequence Service.

Since a Domain may only have one Download Sequence active at any time, the Domain Name is sufficient to identify the Download Sequence being performed. The MMS server shall maintain state information of the Download Sequence as part of the state of the Domain.

If, during the course of a Download Sequence, the association between the client and server is lost, the associated Domain shall be deleted and any partial transfer of information shall be lost. If the association is lost after completion of the Download Sequence, that is when the Domain is in the READY or IN-USE state, the Domain shall be unaffected by the loss of association.

If, during the course of a Download Sequence, any of the service requests is cancelled, the responding MMS-user shall refuse the cancel request unless it is able to maintain the integrity of the Domain. When the processing has progressed to the point that the integrity of the Domain cannot be maintained if cancelled, the cancelable attribute of the Transaction object shall be set to false.

10.1.4.2 Upload Sequence

These services may be used to transfer the Domain Content from the MMS server to the MMS client. Upload is accomplished by the client requesting the InitiateUploadSequence service, zero or more UploadSegment services (as required), and then a TerminateUploadSequence service, in that order.

If, during the course of an Upload Sequence, the association between the client and server is lost, the Upload Sequence shall be terminated and the associated Upload State Machine (ULSM) shall be deleted. The Domain shall be unaffected. Each successful InitiateUploadSequence service invocation shall create an ULSM which is identified by a unique (among all active ULSMs on the association) ULSM ID. The ULSM shall be created and the ULSM ID assigned at the time of the InitiateUploadSequence by the MMS server. An ULSM may only be referenced via the assigned ULSM ID, and only over the association through which it was assigned. The ULSM shall be deleted and the ULSM ID released via the TerminateUploadSequence service, or when the association is aborted.

If, during the course of an Upload Sequence, any of the service requests is cancelled, the responding MMS-user shall refuse the cancel request unless it is able to maintain the integrity of the Upload State Machine. When the processing has progressed to the point that the integrity of the state machine cannot be maintained if cancelled, the cancelable attribute of the Transaction object shall be set to false.

The MMS services for uploading a Domain are interdependent. This interdependence is specified by the Upload State Machines (ULSM) given in Figure 5.

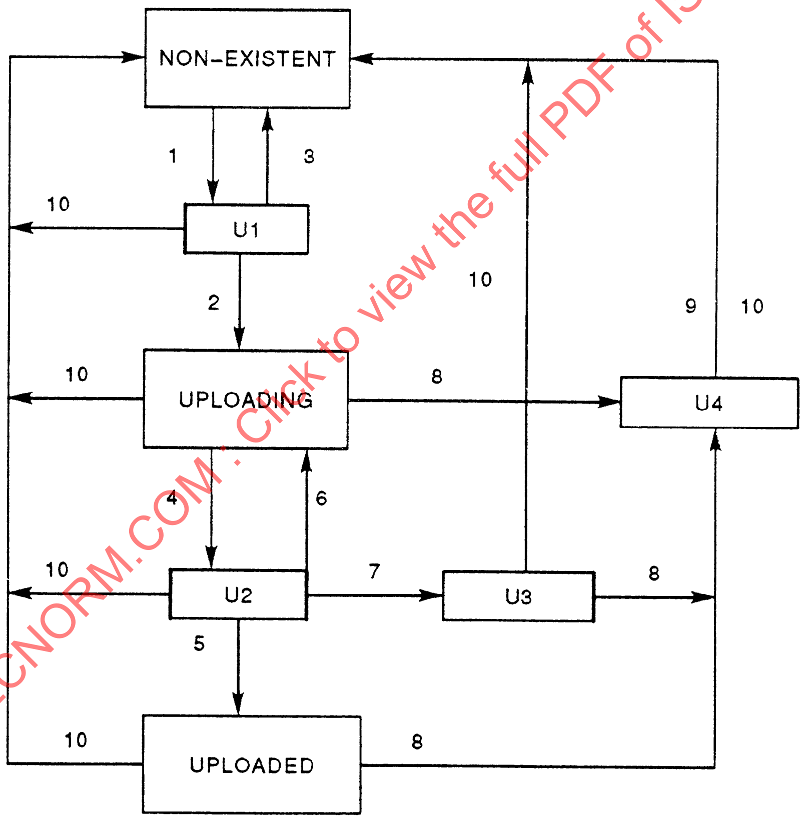


Figure 5 – Upload State Machines

Key:
1 - InitiateUploadSequence.indication
2 - InitiateUploadSequence.response(+)
3 - InitiateUploadSequence.response(-)
4 - UploadSegment.indication

5 - UploadSegment.response(+) moreFollows = false
6 - UploadSegment.response(+) moreFollows = true
7 - UploadSegment.response(-)
8 - TerminateUploadSequence.indication
9 - TerminateUploadSequence.response
10 - Abort.indication or Abort.request

10.2 InitiateDownloadSequence Service

The InitiateDownloadSequence service may be requested by the MMS client to instruct the MMS server to create the named Domain and to begin its loading.

10.2.1 Structure

The structure of the component service primitives is shown in Table 16.

Table 16 – InitiateDownloadSequence Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
List Of Capabilities		M	M(=)			
Sharable		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

10.2.1.1 Argument

This parameter shall convey the parameters of the InitiateDownloadSequence service request.

10.2.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain (at the MMS server) which is to be downloaded.

10.2.1.1.2 List Of Capabilities

This parameter, of type character string, shall represent an implementation specific limitation on the resources of the VMD which are to be part of this Domain. The list of capabilities becomes a defining element of the Domain. If the list of capabilities is not valid and available within the resources of the VMD, an Error Response shall be returned. The determination of valid and available is a local matter.

NOTE – The only capabilities which need to be included are those which must be specified in order that the MMS server can properly perform this service request. It is preferable that this parameter not be used at all, since this promotes the greatest degress of inter- operability. To indicate this, a list with zero elements should be specified.

10.2.1.1.3 Sharable

This parameter, of type boolean, shall specify if true that following loading the Domain may be used by more than one Program Invocation concurrently. Such Domains are said to be sharable. The value false shall specify that the Domain may be used by only one Program Invocation.

10.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. The presence of this parameter shall indicate that the MMS server is ready to receive the load data.

10.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

10.2.2 Service Procedure

The MMS server shall verify that no Domain of the specified name exists, and that the list of capabilities is valid and available. If these conditions are met, the MMS server shall create a suitable Domain with its MMS Deletable attribute set to true, and place it in the LOADING state. The List of Capabilities attribute shall be set to the value of the List of Capabilities parameter of the service request. The Assigned Application Association attribute of the Domain shall be set to reference the application association over which the request service primitive was received. The Sharable attribute of the Domain shall be set to the value of the Sharable parameter of the service request. The List of Program Invocation References shall be initialized to a list with zero elements. The Upload In Progress attribute shall be initialized to zero. The Domain Content and List of Subordinate Objects shall be initialized to be empty. The MMS server shall perform any other actions necessary to prepare for the Download Sequence, and then shall issue a Result(+) service primitive. If the above conditions are not met, the MMS server shall issue a Result(-) service primitive.

10.3 DownloadSegment Service

This service shall be requested by the MMS server to request that a segment of download information be transferred by the MMS client.

10.3.1 Structure

The structure of the component service primitives is shown in Table 17.

Table 17 – DownloadSegment Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Load Data				M	M(=)	
More Follows				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.3.1.1 Argument

This parameter shall convey the parameter of the DownloadSegment service request.

10.3.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the Domain which is to be loaded. The Domain shall be in the LOADING state.

10.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following service specific parameters.

10.3.1.2.1 Load Data

This parameter shall contain the information to be downloaded. This parameter shall be either an octet string or an externally encoded value. The MMS server shall use this information to construct the Domain Content. As part of this process, the MMS server shall create and assign values to all the subordinate objects of this Domain. MMS does not provide transformation services for this data.

NOTE – MMS makes no requirements regarding the nature of the information in Load Data. Load Data may have been created as a result of an Upload Sequence (see 10.5) or as a result of a target device specific programming function (such as an APT post-processor).

10.3.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether (true) or not (false) any additional load data remains to be transmitted for the Domain named in the Download Sequence. This parameter shall be false if Load Data is a zero length string.

10.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.3.2 Service Procedure

The responding MMS-user shall prepare load data for transmission to the requesting MMS-user. The procedure used to create and segment the load data shall be a local matter. If there is such load data to transmit, it shall become the value of the Load Data parameter and the More Follows parameter shall be set equal to true or false, according to whether there remains more load data to be transmitted on a subsequent DownloadSegment service response. If there is no more load data to transmit, the responding MMS-user shall issue a Result(+) with a zero length string as the value of the Load Data Parameter and the More Follows parameter equal to false. If the responding MMS-user detects an error which invalidates the content of the Domain being downloaded, it shall issue a Result(-) response to the service request.

The requesting MMS-user shall receive the load data segments and interpret the load data according to a Domain-specific format, storing them as appropriate. If More Follows is false, the Domain shall be placed in the COMPLETE state following completion of this service. If More Follows is true, the Domain shall remain in the LOADING state.

10.4 TerminateDownloadSequence Service

The TerminateDownloadSequence service shall be requested by the MMS server to indicate to the MMS client that the Download Sequence is complete.

10.4.1 Structure

The structure of the component service primitives is shown in Table 18.

Table 18 – TerminateDownloadSequence Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
Discard		C	C(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.4.1.1 Argument

This parameter shall convey the parameters of the TerminateDownloadSequence service request.

10.4.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the Domain whose Download Sequence is to be terminated. The Domain shall be in the LOADING, COMPLETE, or INCOMPLETE state.

10.4.1.1.2 Discard

This parameter, of type ServiceError, shall indicate if present that the downloaded Domain has been deleted. If the Domain has been deleted, this parameter shall provide an indication of the problem encountered. In that case, the Download Sequence shall have been aborted and the MMS server shall have discarded any portion of the Domain Content which has been received. Otherwise, the downloaded Domain Content shall be retained and the Domain shall be placed in the READY state.

10.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

10.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.4.2 Service Procedure

The MMS server shall request the TerminateDownloadSequence service following completion of the Download Sequence or following detection of an unrecoverable error (by either the client or the server) during the download process.

If the MMS server has detected an unrecoverable error in the course of the Download Sequence, it shall provide the Discard parameter describing the error. If the MMS client has provided a Result(-) to a DownloadSegment request (10.3.2), the MMS server shall provide a Discard parameter indicating that the error was detected by the MMS client. If the server has successfully completed the Download Sequence (and therefore has not set the Discard parameter) and the MMS client indicates success by returning a Result(+) to the TerminateDownloadSequence request, the Download Sequence has succeeded. Otherwise the Download Sequence shall have failed.

If the Download Sequence has succeeded, the Domain shall be placed in READY state. If the Download Sequence has failed, any load data shall be discarded and the Domain shall be deleted.

10.5 InitiateUploadSequence Service

The MMS client may request the InitiateUploadSequence service to instruct the MMS server to prepare to upload the Domain of the specified name.

10.5.1 Structure

The structure of the component service primitives is shown in Table 19.

Table 19 – InitiateUploadSequence Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
ULSM ID				M	M(=)	
List Of Capabilities				M	M(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

10.5.1.1 Argument

This parameter shall convey the parameter of the InitiateUploadSequence service request.

10.5.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain whose content is to be transferred to the MMS client (uploaded).

10.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. Returning this parameter shall also indicate confirmation that the MMS server is ready to participate in the Upload Sequence.

10.5.1.2.1 ULSM ID

This parameter, of type integer, shall specify the ULSM object which is created as a result of this request.

10.5.1.2.2 List Of Capabilities

This parameter, of type list of character string, shall identify the list of capabilities which were used in the creation or definition of this Domain.

NOTE – The only capabilities which need to be included are those which must be specified in order that the MMS server can properly perform this service. It is preferable that this parameter not be used at all, since this promotes the greatest degree of interoperability. To indicate this, a list with zero elements should be specified.

10.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.5.2 Service Procedure

Upon receipt of the InitiateUploadSequence indication primitive, the MMS server shall verify that the specified Domain is in the READY or the IN-USE state. If this condition is met, the MMS server shall create a ULSM and assign a unique integer to it. The Upload In Progress attribute of the Domain shall be increased by 1. The MMS server shall take whatever other actions are necessary to prepare to upload the specified Domain.

NOTE – If an Upload Sequence is undertaken for a Domain which is in the IN-USE state, the Domain Content may be changing during the time the Upload Sequence is being performed. This may result in incomplete or inconsistent data being uploaded. Interpretation of the upload in this situation is a local matter.

10.6 UploadSegment Service

This service may be invoked by the MMS client to request the transfer of a segment of upload data from the specified Domain by the MMS server.

10.6.1 Structure

The structure of the component service primitives is shown in Table 20.

Table 20 – UploadSegment Service

Table 20 (Cont.) – UploadSegment Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
ULSM ID		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Load Data				M	M(=)	
More Follows				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.6.1.1 Argument

This parameter shall convey the parameter of the UploadSegment service request.

10.6.1.1.1 ULSM ID

This parameter, of type integer, shall specify the instance of the ULSM which controls this transfer. The Domain to be uploaded is implicitly identified by this parameter. If this parameter does not match an active ULSM, a Result(-) response shall be returned.

10.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. In this case, the following two parameters shall also be returned.

10.6.1.2.1 Load Data

This parameter shall contain the requested load data from the MMS server. This parameter shall be either an octet string or an externally encoded value.

10.6.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether (true) or not (false) more load data remains to be transferred to complete the Upload Sequence. This parameter shall be false if Load Data is a zero length string.

10.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.6.2 Service Procedure

The MMS server shall provide the content of each upload segment such that it is formatted for receipt as load data in a later DownloadSegment (download) service. If the load data cannot be placed in this format, an Result(-) response shall be returned. If the end of the load data has been reached in this sequence, the MMS server shall return More Follows equals false as part of its response.

10.7 TerminateUploadSequence Service

An MMS client may use the TerminateUploadSequence service to request that the MMS server terminate an Upload Sequence. In particular, the TerminateUploadSequence service causes the corresponding ULSM to be deleted, whether or not the service completed or was successful.

10.7.1 Structure

The structure of the component service primitives is shown in Table 21.

Table 21 – TerminateUploadSequence Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
ULSM ID		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.7.1.1 Argument

This parameter shall convey the parameter of the TerminateUploadSequence service request.

10.7.1.1.1 ULSM ID

This parameter, of type integer, shall identify the instance of the ULSM whose Upload Sequence is to be terminated. If this value does not match a valid instance of an existing ULSM, a Result(-) response shall be returned.

10.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded, implying that the Upload Sequence has terminated successfully. A successful result does not supply service specific parameters.

10.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.7.2 Service Procedure

The MMS client shall request the TerminateUploadSequence service following completion of the Upload Sequence or following an error reported by the MMS server in a Result(-) response to a UploadSegment service. The Upload In Progress attribute of the Domain shall be decreased by one (1). Following either the successful or unsuccessful completion of this service, the ULSM shall be deleted. If the MMS server detects a problem in the Upload Sequence, it shall return an Result(-) response. Such a response shall be returned, for instance, if the MMS client requests the TerminateUploadSequence service before the MMS server has returned the More Follows parameter with a value equal to false.

10.8 RequestDomainDownload Service

The RequestDomainDownload service may be used to request that the responding MMS-user initiate a Download Sequence with the requesting MMS-user.

10.8.1 Structure

The structure of the component service primitives is shown in Table 22.

Table 22 – RequestDomainDownload Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
List Of Capabilities		M	M(=)			
Sharable		M	M(=)			
File Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.8.1.1 Argument

This parameter shall convey the parameters of the RequestDomainDownload service request.

10.8.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain which is to be downloaded.

10.8.1.1.2 List Of Capabilities

This parameter, of type list of character string, shall be used in the subsequent InitiateDownloadSequence service request.

10.8.1.1.3 Sharable

This parameter, of type boolean, shall indicate if true that the Domain may be used by multiple Program Invocations. Such Domains are said to be sharable. Otherwise, the Domain may be used by only one Program Invocation.

10.8.1.1.4 File Name

This parameter, of type FileName, shall specify the name of the file (as known by the responding MMS-user) containing the information to be loaded. If the named file does not exist or cannot be accessed, a Result(-) response shall be returned.

10.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded, confirming that the specified file has been loaded. A successful result does not supply service specific parameters.

10.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.8.2 Service Procedure

The responding MMS-user shall perform the RequestDomainDownload service by requesting the InitiateDownloadSequence service as described in 10.2. The values of the Domain Name, List Of Capabilities and Sharable parameters received in the indication service primitive shall be used as the values of the parameters of the same name for the InitiateDownloadSequence service. The value of the File Name parameter shall be used as the source of the load data.

If the load is completed successfully, then the Result(+) parameter in the response primitive shall indicate success of the RequestDomainDownload service. If any element of the Download Sequence returns a Result(-) response, that response shall be reflected in a Result(-) response to the RequestDomainDownload service.

NOTE – A request to cancel a RequestDomainDownload service may require very complex processing if the procedure has progressed to the point where a Domain has been created. It is a local matter when to set the cancelable attribute of the RequestDomainDownload Transaction Object to false.

10.9 RequestDomainUpload Service

This service allows an MMS-user to request that the contents of a specified Domain located at the requesting MMS-user be uploaded to the responding MMS-user.

10.9.1 Structure

The structure of the component service primitives is shown in Table 23.

Table 23 – RequestDomainUpload Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
File Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.9.1.1 Argument

This parameter shall convey the parameters of the RequestDomainUpload service request.

10.9.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain whose contents are to be uploaded.

10.9.1.1.2 File Name

This parameter, of type FileName, shall specify the file name as known by the responding MMS-user to be used to store the Domain upload. If this file cannot be accessed, a Result(-) response shall be returned.

10.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

10.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.9.2 Service Procedure

The responding MMS-user shall perform the Upload Sequence as described in 10.5 and store the resulting load data in the named file. If the service request in the Upload Sequence returns a Result(-) response, that Result(-) response shall be reflected in a Result(-) response to the RequestDomainUpload service request.

NOTE – A request to cancel a RequestDomainUpload service may require very complex processing if the procedure has progressed to the point where an Upload State Machine has been created. It is a local matter when to set the cancelable attribute of the RequestDomainUpload Transaction Object to false.

10.10 LoadDomainContent Service

The LoadDomainContent service may be used by the MMS client to request that the MMS server load a file from its own filestore or from a third party into a designated Domain. A typical sequence of operations involving a third party which employs MMS services is shown in Figure 6:

10.10.1 Structure

The structure of the component service primitives is shown in Table 24.

MMS client	MMS server	Third Party
LoadDomainContent request	→	
	RequestDomainDownload request	→
	←	InitiateDownloadSeq request
	InitiateDownloadSeq response	→
	DownloadSegment request	→
	←	DownloadSegment response
	...	
		...
	TerminateDownloadSeq request	→
	←	TerminateDownloadSeq response
	←	RequestDomainDownload response
←	LoadDomainContent response	

Figure 6 – LoadDomainContent

Table 24 – LoadDomainContent Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
List Of Capabilities		M	M(=)			
Sharable		M	M(=)			
File Name		M	M(=)			
Third Party		U	U(=)			TPY
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.10.1.1 Argument

This parameter shall convey the parameters of the LoadDomainContent service request.

10.10.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain which is to be loaded.

10.10.1.1.2 List Of Capabilities

This parameter, of type list of character string, shall indicate the capabilities to be used in the creation of the named Domain.

10.10.1.1.3 Sharable

This parameter, of type boolean, shall indicate if true that the Domain can be used by several Program Invocations concurrently. Otherwise, the Domain can be used by only one Program Invocation.

10.10.1.1.4 File Name

This parameter, of type FileName, shall specify the name of the file containing the information to be loaded.

10.10.1.1.5 Third Party

This parameter, of type ApplicationReference, shall specify the application reference of the Application Process through which the named file may be accessed. Support of processing for this parameter is an implementation option which shall be implemented if support for the TPY parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the MMS server shall attempt to access the requested file directly.

10.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded, confirming that the specified file has been loaded. A successful result does not supply service specific parameters.

10.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.10.2 Service Procedure

The responding MMS-user shall perform the LoadDomainContent Service as follows:

a) verify that no Domain of that name exists;

b) create and initialize a Domain following the service procedure described in 10.2.2.

c) if a third party is specified, establish an association with that application if none exists; thereafter take appropriate action to cause the named Domain to be obtained and loaded.

d) if no third party is specified, perform the necessary steps to obtain the file through local means and load it into the specified Domain.

e) if loading is successful, place the Domain in the READY state.

If a third party is specified and the responding MMS-user is unable to establish or maintain an association with the third party, a Result(-) response shall be returned.

If the load is completed successfully, a Result(+) response primitive shall be issued. If the load is not completed successfully, a Result(-) response primitive shall be issued.

NOTE – A request to cancel a LoadDomainContent service may require very complex processing if the procedure has progressed to the point where a Domain has been created. It is a local matter when to set the cancelable attribute of the LoadDomainContent Transaction Object to false.

10.11 StoreDomainContent Service

This service allows an MMS client to request that the contents of a specified Domain at the MMS server be stored in a file on a filestore. "Storing" a Domain requires whatever processing is necessary such that it may be loaded later using the LoadDomainContent service. A typical sequence of operations involving a third party which employs MMS services is shown below in Figure 7:

10.11.1 Structure

The structure of the component service primitives is shown in Table 25.

MMS client	MMS server	Third Party
StoreDomainContent request	→	
	RequestDomainUpload request	→
	←	InitiateUploadSeq request
	InitiateUploadSeq response	→
	←	UploadSegment request
	UploadSegment response	→
	...	
		...
	←	TerminateUploadSeq request
	TerminateUploadSeq response	→
	←	RequestDomainUpload response
←	StoreDomainContent response	

Figure 7 – StoreDomainContent

Table 25 – StoreDomainContent Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
File Name		M	M(=)			
Third Party		U	U(=)			TPY
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.11.1.1 Argument

This parameter shall convey the parameters of the StoreDomainContent service request.

10.11.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain whose contents are to be stored to a file.

10.11.1.1.2 File Name

This parameter, of type FileName, shall identify the file in which the Domain is to be stored.

10.11.1.1.3 Third Party

This optional parameter, of type ApplicationReference, shall identify the third party on which the filestore resides which is to receive the contents of the named Domain. The presence of this parameter is an implementation option which shall be implemented if support of the TPY parameter conformance building block is claimed. If TPY is implemented, its use is a user option. If this parameter is absent, the responding MMS-user shall attempt to store the content of the Domain directly in the specified file.

10.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

10.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.11.2 Service Procedure

If a Third Party is specified, the responding MMS-user shall establish an association with that application if none exists. If a Third Party is specified and the responding MMS-user is unable to establish or maintain an association with the third party, a Result(-) response shall be returned. The responding MMS-user shall take appropriate action to cause the contents of the Domain to be stored in the indicated file.

If the upload is completed successfully, then the result parameter in the response primitive shall be provided and shall indicate success of the StoreDomainContent service. If the upload is not completed successfully, then the Result(-) response shall be returned.

NOTE – A request to cancel a StoreDomainContent service may require very complex processing if the procedure has progressed to the point where an Upload State Machine has been created. It is a local matter when to set the cancelable attribute of the StoreDomainUpload Transaction Object to false.

10.12 DeleteDomain Service

The DeleteDomain service may be used to request that an MMS server delete the specified Domain.

10.12.1 Structure

The structure of the component service primitives is shown in Table 26.

Table 26 – DeleteDomain Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Domain Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

10.12.1.1 Argument

This parameter shall convey the parameter of the DeleteDomain service request.

10.12.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain which is to be deleted.

10.12.1.2 Result(+)

The Result(+) parameter shall indicate that the Domain was successfully deleted. A successful result does not supply service specific parameters.

10.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type, which is defined in detail in clause 17, shall provide the reason for failure.

10.12.2 Service Procedure

The MMS server shall verify that the specified Domain exists, that its MMS-Deletable attribute is true, that it is in the READY state, and that there are no uploads in progress. If these conditions are met, the MMS server shall delete the designated Domain. As a part of this action, it shall delete all objects subordinate to the Domain regardless of whether the MMS-Deletable attribute of these objects is true or not, and it shall set the reference attribute of the objects which refer to objects subordinate to this Domain to UNDEFINED. In particular, the Monitored Variable Reference attribute of an Event Condition object which refers to an MMS variable object subordinate to this Domain shall be changed to UNDEFINED (see 15.1.1.1). If the service cannot be performed, a Result(-) response shall be returned.

NOTE – The requirement that the Domain be in the READY state is equivalent to the requirement that no Program Invocation be bound to the Domain. Hence, Domains which are currently in use by some Program Invocation cannot be deleted until the Program Invocation is deleted.

10.13 GetDomainAttributes Service

The GetDomainAttributes service may be used to request that an MMS server return the attributes associated with the specified Domain.

10.13.1 Structure

The structure of the component service primitives is shown in Table 27.

Table 27 – GetDomainAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Domain Name	M	M(=)			
Result(+) (COMP)			S	S(=)	
List Of Capabilities			M	M(=)	
State			M	M(=)	
MMS Deletable			M	M(=)	
Sharable			M	M(=)	
List Of Program Invocations			M	M(=)	
Upload In Progress			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

10.13.1.1 Argument

This parameter shall convey the parameter of the GetDomainAttributes service request.

10.13.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain whose attributes are requested.

10.13.1.2 Result(+)

The Result(+) parameter shall indicate that the Domain exists and that its attributes are provided in the response.

10.13.1.2.1 List Of Capabilities

This parameter, of type list of character string, shall specify the capabilities used in the creation of this Domain.

10.13.1.2.2 State

This parameter, of type DomainState, shall specify whether the Domain is in the LOADING, COMPLETE, INCOMPLETE, READY, IN-USE, or one of the transition states D1 through D9.

10.13.1.2.3 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Domain may be deleted.

10.13.1.2.4 Sharable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Domain may be simultaneously used by more than one Program Invocation.

10.13.1.2.5 List Of Program Invocations

This parameter, of type list of Identifier, shall specify the Program Invocations which are linked to this Domain. If the Domain is not sharable. there shall be at most one such Program Invocation. If the Domain is not in the IN-USE state, in the state D5 or in the state D6, this list shall be empty.

10.13.1.2.6 Upload In Progress

This parameter, of type integer, shall indicate the number of Upload Sequences currently active for this Domain. The value zero indicates that no upload is in progress.

10.13.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

10.13.2 Service Procedure

The MMS server shall verify that the specified Domain exists. It shall return the values of the attributes of the Domain as values of the corresponding Result(+) parameters. If the Domain does not exist a Result(-) response shall be returned.

NOTE – The List Of Subordinate Objects attribute is not included in the response. The names of these objects may be obtained using the GetNameList Service, specifying Domain-specific scope and naming this Domain (see 9.4).

11 Program Invocation Management Services

The MMS model of the Virtual Manufacturing Device (VMD) described in clause 7 introduces four abstract elements: Executive function, Capabilities, Program Invocations, and Domains. This clause covers the operations on Program Invocations. Program Invocations may be dynamic in nature, coming into existence and being removed from the system either by MMS services or by local action. Program Invocations may also be predefined within the VMD. Services are provided to allow an MMS client to affect the behaviour of Program Invocations.

11.1 The Program Invocation Object

A Program Invocation is a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment. It is either predefined or created, either by MMS services or by local action. It is composed of a set of Domains together with control information necessary for its execution. The execution of a Program Invocation can be thought of as a time series of fundamental operations of the underlying device. Further definition of these fundamental operations is a local matter. In this time history of a Program Invocation, there is one distinguished state in which the Program Invocation is ready for execution but has not yet begun execution. This state is referred to as the IDLE state of the Program Invocation. Some Program Invocations also have a distinguished state in which the Program Invocation has "completed". Program Invocations which reach this state have accomplished their purpose and cannot be placed in execution again. This state is called UNRUNNABLE, and Program Invocations which normally reach this state are called non-reusable. Program Invocations which are reusable will return to the IDLE state following the normal completion of their execution.

11.1.1 The Program Invocation Object Model

Object: Program Invocation

```
Key Attribute: Program Invocation Name
Attribute: State (IDLE,STARTING,RUNNING,STOPPING,STOPPED,
                  RESUMING, RESETTING, UNRUNNABLE)
Attribute: List Of Domain References
Attribute: MMS Deletable(TRUE,FALSE)
Attribute: Reusable (TRUE,FALSE)
Attribute: Monitor  (TRUE, FALSE)
Constraint: Monitor = TRUE
     Attribute: Event Condition Reference
     Attribute: Event Action Reference
     Attribute: Event Enrollment Reference
Attribute: Execution Argument
Attribute: Additional Detail
```

Program Invocation Name

The Program Invocation Name shall be the principal identifier of the Program Invocation. It shall be formed according to the rules for the specification of Object Names as described in 7.4 and 7.3.2.

State

The State attribute shall indicate the principal states of the Program Invocation. In order to complete state diagrams, the value NON-EXISTENT is added to this list to describe the condition before a Program Invocation is created.

NOTE 1 – The State attribute is the only element of the control information necessary for the execution of this Program Invocation which is defined by this part of ISO/IEC 9506. Companion Standards may define representations for additional elements of the control information as appropriate for their application areas. In particular, for Program Invocations which are implemented through a sequential procedural programming language, the concepts of "beginning of program", "end of program" and "program step" may be added to the control information.

IDLE

This state shall denote the condition of a Program Invocation at a time before it is placed in operation.

NOTE 2 – If the Program Invocation is implemented through a sequential procedural programming language, this state may correspond to the "beginning of the program".

RUNNING

This state shall denote the condition of a Program Invocation during its execution. Further definition of "execution" is a local matter. However, RUNNING is usually associated with a process which changes the constituent elements of at least one of its subordinate Domains.

STOPPED

This state shall denote a condition in which the Program Invocation is at an intermediate state between the onset of execution and completion. However, execution has ceased and the constituent elements of the subordinate Domains are no longer changing due to the action of this Program Invocation.

UNRUNNABLE

This state shall denote a condition in which the Program Invocation may no longer be executed, but has not yet been deleted. This state may be reached by the completion of the Program Invocation if the Reusable attribute is false or through explicit MMS service action or through other local action.

STARTING

This state shall be a transitory state of the Program Invocation between the IDLE and the RUNNING states. If the Program Invocation is placed in the RUNNING state through the action of the MMS Start service, this state corresponds to the time between the receipt of the Start service indication primitive and the issuance of the Start service response primitive. While in the STARTING state, the VMD may perform one or more initialization procedures which may be locally defined or may be required by Companion Standards.

STOPPING

This state shall be a transitory state of the Program Invocation between the RUNNING and the STOPPED states. If the Program Invocation is placed in the STOPPED state through the action of the MMS Stop service, this state corresponds to the time between the receipt of the Stop service indication primitive and the issuance of the Stop service response primitive. While in the STOPPING state, the VMD may perform one or more stopping procedures which may be locally defined or may be required by Companion Standards.

RESUMING

This state shall be a transitory state of the Program Invocation between the STOPPED and the RUNNING states. If the Program Invocation is placed in the RUNNING state through the action of the MMS Resume service, this state corresponds to the time between the receipt of the Resume service indication primitive and the issuance of the Resume service response primitive. While in the RESUMING state, the VMD may perform one or more resumption procedures which may be locally defined or may be required by Companion Standards.

RESETTING

This state shall be a transitory state of the Program Invocation between the STOPPED state and the IDLE state. If the Program Invocation is placed in the IDLE state through the action of the MMS Reset service, this state corresponds to the time between the receipt of the Reset service indication primitive and the issuance of the Reset service response primitive. While in the RESETTING state, the VMD may perform one or more resetting procedures which may be locally defined or may be required by Companion Standards.

List Of Domain References

This attribute shall be a list of references to the Domains which compose this Program Invocation. The list shall contain at least one entry.

MMS Deletable

This attribute shall indicate whether (true) or not (false) the Program Invocation can be deleted using the DeleteProgram-Invocation service.

Reusable

This attribute shall indicate whether (true) or not (false) the Program Invocation will return to the IDLE state following normal completion of its execution. If false, the Program Invocation will move to the UNRUNNABLE state following normal execution.

Monitor

This attribute shall indicate whether or not program monitoring is in effect for this Program Invocation. A Program Invocation which is being monitored uses the Event Management facilities of MMS to inform the requesting MMS-user whenever the Program Invocation leaves the RUNNING state. It does this by creating the referenced objects listed below.

Event Condition Reference

If the Monitor attribute is true, the Program Invocation shall have an attribute which is a reference to an Event Condition object (see 15.1.1) with the following attributes:

a) the Event Condition Name attribute value shall be equal to the Program Invocation object's Program Invocation Name attribute value;

b) the Event Condition Class attribute value shall be equal to MONITORED;

c) the State attribute shall be IDLE;

d) the Enabled attribute shall be true;

e) Monitored Variable Reference shall be equal to UNSPECIFIED (specifying that the Program Invocation is in the RUNNING state);

f) the List Of Event Enrollment Reference attribute shall contain the reference to the following defined Event Enrollment object;

g) the MMS Deletable attribute shall be true;

h) the Priority attribute shall be normalPriority;

i) the Severity attribute shall be normalSeverity;

j) the Alarm Summary Reports attribute shall be false;

k) the Evaluation Interval attribute shall be a local matter.

Event Action Reference

If the Monitor attribute is true, the Program Invocation has an attribute which is a reference to an Event Action object (see 15.1.2) with the following attributes:

a) the Event Action Name attribute shall be equal to the Program Invocation object's Program Invocation Name attribute;

b) the MMS Deletable attribute shall be true;

c) the Confirmed Service Request attribute is the GetProgramInvocationAttributes service;

d) the List Of Modifier attribute shall be empty;

e) the List Of Event Enrollment Reference attribute shall contain a reference to the single Event Enrollment which follows.

Event Enrollment Reference

If the Monitor attribute is true, the Program Invocation shall have an attribute which is a reference to an Event Enrollment (see 15.1.3) with the following attributes:

a) the Event Enrollment Name attribute shall be equal to the Program Invocation object's Program Invocation Name attribute;

b) the MMS Deletable attribute value shall be true;

c) the Event Enrollment Class attribute shall be NOTIFICATION;

d) the Event Condition Reference attribute references the Event Condition described above;

e) the Event Condition Transitions attribute shall contain the sole element ACTIVE-TO-IDLE;

f) the Application Association Local Tag attribute shall specify the association over which the Program Invocation was created;

g) the Notification Lost attribute shall be false;

h) the Event Action Reference attribute shall reference the Event Action described above;

i) the Duration attribute shall be determined by the corresponding parameter of the CreateProgramInvocation service;

j) the Client Application attribute shall reference the client application which has invoked the CreateProgramInvocation service;

k) the Alarm Acknowledgement Rule attribute shall be NONE;

l) the State attribute shall be IDLE.

Execution Argument

This attribute shall contain a character string or an externally coded parameter appropriate to the execution of this Program Invocation. It may be set by either the Start service or by the Resume service. It shall initially be set to a empty string when the Program Invocation is created. Subsequent execution of the Program Invocation may change the value of this attribute.

Additional Detail

This attribute shall allow for additional attributes to be added by Companion Standards.

11.1.2 Operations on Program Invocations

The services which operate on Program Invocations objects are listed below.

CreateProgramInvocation

A client may use this service to create a new Program Invocation object at a VMD. This service is described in 11.2.

DeleteProgramInvocation

A client may use this service to delete a Program Invocation object at a VMD. This service is described in 11.3.

Start

A client may use this service to cause a previously defined Program Invocation which is in the IDLE state to transition into the RUNNING state. This service is described in 11.4.

Stop

A client may use this service to cause a Program Invocation which is in the RUNNING state to transition to the STOPPED state. This service is described in 11.5.

Resume

A client may use this service to cause a Program Invocation which is in the STOPPED state to transition to the RUNNING state. This service is described in 11.6.

Reset

A client may use this service to cause a Program Invocation which is in the STOPPED state to transition either to the IDLE state or to the UNRUNNABLE state. This service is described in 11.7.

Kill

A client may use this service to terminate a Program Invocation by causing it to transition to the UNRUNNABLE state. This service is described in 11.8.

GetProgramInvocationAttributes

A client may use this service to determine the attributes of a Program Invocation, its state, and its list of Domain references. This service is described in 11.9.

11.1.3 Program Invocation State Diagram

NOTE – In Figure 8, some intermediate states (states which exist only between an indication primitive and a response primitive) may be reported as proper states of the Program Invocation. This is because, in general, the transition may take an appreciable time. The states named "Pn" are not reported in the GetProgramInvocationAttributes service since the CreateProgramInvocation and the DeleteProgramInvocation services are defined as atomic (non-interruptible) and therefore appear to be instantaneous to the MMS client. In order to keep the diagram simple, the intermediate states of the Kill service are not shown. These intermediate states are not reported since the Kill service procedure is defined as atomic (non-interruptible), and therefore its effect appears to be instantaneous. The effects of MMS service requests which do not change the state of the Program Invocation, such as GetProgramInvocationAttributes, are not shown on the diagram. The terms destructive and non-destructive are explained in the service procedure for the Cancel service (8.5) and in the respective service procedures described in this clause.

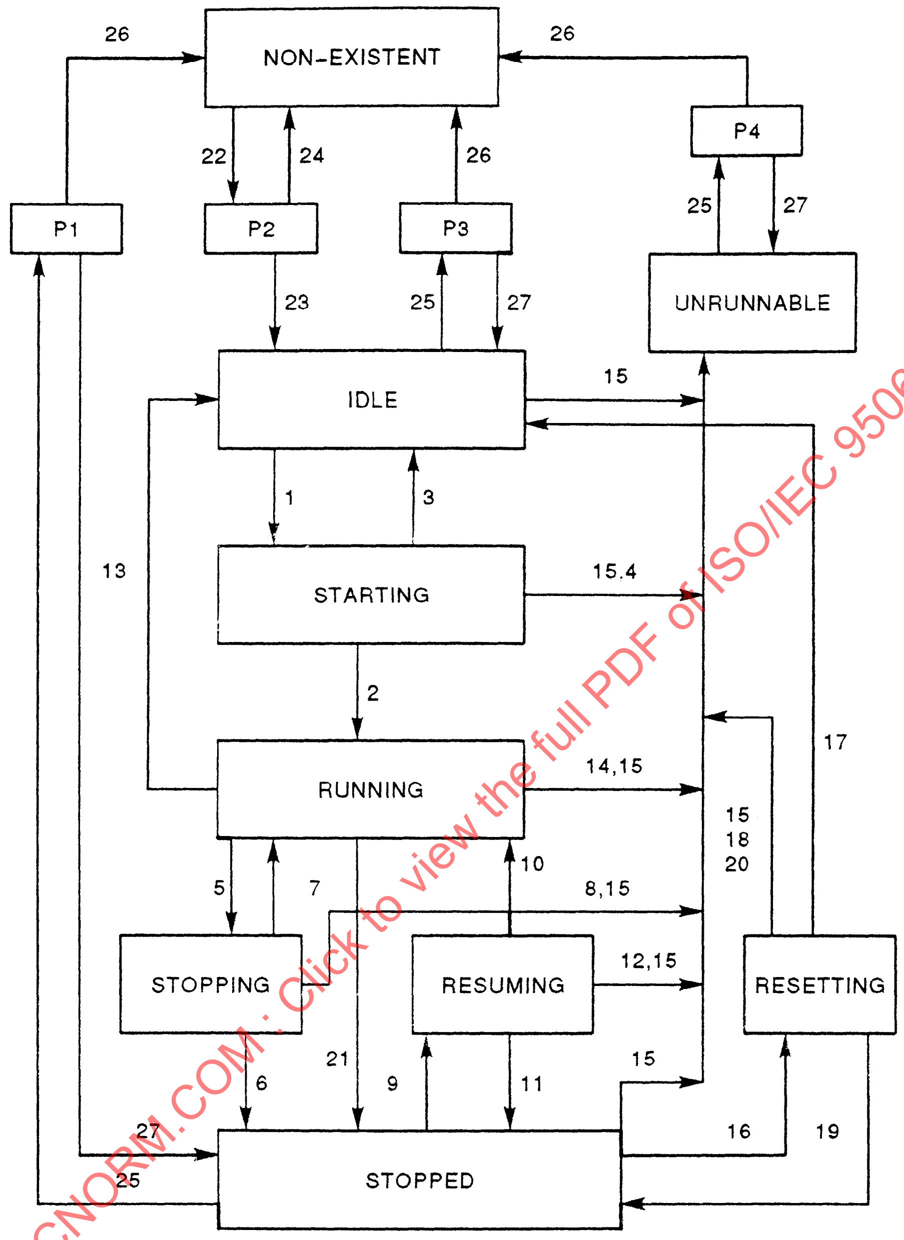


Figure 8 – Program Invocation State Diagram

Transition lines for the model are:

1 - Start.indication
2 - Start.response(+)
3 - Start.response(-) non-destructive
4 - Start.response(-) destructive
5 - Stop.indication
6 - Stop.response(+)
7 - Stop.response(-) non-destructive
8 - Stop.response(-) destructive
9 - Resume.indication

10 - Resume.response(+)
11 - Resume.response(-) non-destructive
12 - Resume.response(-) destructive
13 - (end of program) Reusable = true
14 - (end of program) Reusable = false
15 - Kill.response(+)
16 - Reset.indication
17 - Reset.response(+) Reusable = true
18 - Reset.response(+) Reusable = false
19 - Reset.response(-) non-destructive
20 - Reset.response(-) destructive
21 - (program stop)
22 - CreateProgramInvocation.indication
23 - CreateProgramInvocation.response(+)
24 - CreateProgramInvocation.response(-)
25 - DeleteProgramInvocation.indication
26 - DeleteProgramInvocation.response(+)
27 - DeleteProgramInvocation.response(-)

NOTE – The transition Kill.indication is not included in this diagram because the action of the Kill service is atomic; there is no difference in the state of the Program Invocation between the indication and the response(+). The Kill.response(-) is not included because it does not cause a state transition.

11.2 CreateProgramInvocation Service

The CreateProgramInvocation service allows an MMS client to assemble Domains into a specified Program Invocation at the MMS server. The MMS client specifies a list of Domains which are to be included in the Program Invocation. Note that a given Domain may be in the List of Domain Reference of more than one Program Invocation simultaneously if it is sharable.

11.2.1 Structure

The structure of the component service primitive is show in Table 28.

Table 28 – CreateProgramInvocation Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Program Invocation Name		M	M(=)			
List Of Domain Names		M	M(=)			
Resusable		M	M(=)			
Monitor		U	U(=)			
Monitor Type		C	C(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

11.2.1.1 Argument

This parameter shall contain the parameters of the CreateProgramInvocation service request.

11.2.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the name to be associated with this Program Invocation. If this name is already in use, a Result(-) response shall be returned.

11.2.1.1.2 List Of Domain Names

This parameter, of type list of Identifier, shall specify the existing Domains by name which are to be incorporated as part of this Program Invocation. There shall be at least one such element in this list. The order of the list may be significant to the MMS server in the process of creating the Program Invocation.

11.2.1.1.3 Reusable

This parameter, of type boolean, shall indicate whether (true) the Program Invocation will enter the IDLE state after completion or (false) the UNRUNNABLE state.

11.2.1.1.4 Monitor

This parameter, of type boolean, shall indicate, if present, that the MMS client wishes to be informed about the progress of the Program Invocation as it is executed.

11.2.1.1.4.1 Monitor Type

This parameter, of type boolean, shall be present if and only if the value of the Monitor parameter is true. This parameter shall indicate, if true, that the notification is PERMANENT, hence the notification exists for the life of the Program Invocation. If the value is false, the notification is CURRENT, and exists only as long as the association is maintained. The value of this parameter shall become the value of the Duration attribute of the Event Enrollment which is implicitly created as a result of this service request.

11.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. In this case, no parameters are returned.

11.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides a reason for failure.

11.2.2 Service Procedure

The MMS server shall perform the following series of actions:

a) verify that no Program Invocation of the same name already exists;

b) verify the existence of all Domains of the parameter list; if a listed Domain does not exist, an error shall be returned.

c) verify that each Domain is available for incorporation into this Program Invocation (that each Domain is either in the READY or D7 state, or is in the IN-USE, D4, D5, or D6 state with Sharable attribute equal to true);

d) create a ProgramInvocation object with its Program Invocation Name attribute equal to the Program Invocation Name parameter, the State attribute equal to IDLE, the MMS Deletable attribute equal to true, the Reusable attribute equal to the Reusable parameter, the Monitor attribute equal to true if the Monitor parameter is specified, otherwise false, and the Execution Argument attribute equal to a string of length zero;

e) place references to the Domains on the Program Invocation's List Of Domain Reference attribute;

f) for each Domain in the List of Domain Names parameter, perform the following steps:

   1) if the Domain State attribute is equal to D4, D5, or D6, wait until the Domain enters the IN-USE state;

   2) if the Domain State atribute is equal to D7, wait until the Domain enters the READY state;

   3) change the Domain State attribute to IN-USE;

   4) add a reference to this Program Invocation to the Domain's List of Program Invocation Reference attribute.

g) if the Monitor parameter is specified, enable the Program Invocation to report whenever it ceases running. To do this, it shall make use of Event Management facilities described in clause 15. The following actions shall occur which will cause the Program Invocation to issue a EventNotification request primitive that carries a GetProgramInvocationAttribute response whenever the Program Invocation leaves the RUNNING state (see 11.1.3). The MMS server

   1) shall create an Event Condition with the following attributes:

      i) the Event Condition Name attribute shall be equal to the Program Invocation Name parameter,

      ii) the MMS Deletable attribute shall be true,

      iii) the Event Condition Class attribute value shall be equal to MONITORED,

      iv) the State attribute shall be equal to IDLE,

      v) the Priority attribute shall be equal to normalPriority,

      vi) the Severity attribute shall be equal to normalSeverity,

      vii) the List Of Event Enrollment Reference attribute shall contain a single reference to the Event Enrollment whose Event Enrollment Name attribute is equal to the Program Invocation Name parameter,

      viii) the Enabled attribute shall be true,

      ix) the Alarm Summary Reports attribute shall be false,

      x) the Monitored Variable Reference attribute value shall be equal to UNSPECIFIED specifying that the Program Invocation is in the RUNNING state,

      xi) the Evaluation Interval attribute shall be set to a value which is determined as a local matter;

   2) shall create an Event Action with the following attributes:

      i) the Event Action Name attribute shall be equal to the Program Invocation Name parameter,

      ii) the MMS Deletable attribute shall be true,

      iii) the Confirmed Service Request attribute shall be the GetProgramInvocationAttributes service with a Service Argument equal to the Program Invocation Name parameter,

      iv) the List Of Modifier attribute shall have zero elements,

      v) the List Of Event Enrollment Reference shall contain a single reference to an Event Enrollment whose Event Enrollment Name attribute is equal to the Program Invocation Name parameter;

   3) shall create an Event Enrollment with the following attributes:

      i) the Event Enrollment Name attribute value shall be equal to the Program Invocation Name parameter,

      ii) the MMS Deletable attribute shall be true,

      iii) the Enrollment Class attribute shall be NOTIFICATION,

      iv) the Event Condition Reference attribute shall be a reference to the Event Condition whose Event Condition Name attribute is equal to the Program Invocation Name parameter,

v) the Event Condition Transitions attribute shall contain the single element ACTIVE-TO-IDLE,

vi) the Application Association Local Tag attribute shall specify the current association,

vii) the Notification Lost attribute shall be false,

viii) the Event Action Reference shall be a reference to the Event Action whose Event Action Name attribute is equal to the Program Invocation Name parameter,

ix) the Duration attribute shall be equal to either CURRENT or PERMANENT depending on the value of the Monitor Type parameter,

x) the Client Application attribute shall reference the application process making this request,

xi) the Alarm Acknowledgement Rule attribute shasll be NONE,

xii) the State attribute shall be IDLE.

NOTE – This is equivalent to transitions of the Program Invocation out of the RUNNING state, i.e. whenever it completes or is stopped (locally or via MMS services), or is killed if it was in the RUNNING state.

Additionally, a VMD shall guarantee that this service procedure is atomic, (not interruptible by another MMS service indication specifying this Program Invocation or its constitutive elements).

If any step in this process fails, the service shall fail, all partially completed steps shall be undone, and a Result(-) response shall be returned. Otherwise, the service shall succeed and the Program Invocation shall be left in IDLE state.

11.3 DeleteProgramInvocation Service

The DeleteProgramInvocation service allows an MMS client to cause the deletion of an existing Program Invocation at the MMS server.

11.3.1 Structure

The structure of the component service primitives is shown in Table 29.

Table 29 – DeleteProgramInvocation Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Program Invocation Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

11.3.1.1 Argument

This parameter shall contain the parameter of the DeleteProgramInvocation service request.

11.3.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the name of the Program Invocation which is to be deleted.

11.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. There are no service specific parameters.

11.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.3.2 Service Procedure

The MMS server shall verify that the specified Program Invocation exists, that its MMS Deletable attribute is true, and that it is in the IDLE, STOPPED, or UNRUNNABLE state. If these conditions are not met a Result(-) response shall be returned. If these conditions are met, the MMS server shall remove references to the Program Invocation from the List Of Program Invocation Reference attributes of each Domain referenced by the Program Invocation. If the subject Domain has no other references to Program Invocations, it shall be moved from the IN-USE state to the READY state. If the Program Invocation object's Monitor attribute value is true, the associated Event Enrollment, Event Action, and Event Condition shall be deleted according to the procedures described in the DeleteEventEnrollment, DeleteEventAction and DeleteEventCondition service procedures, except that the value of the MMS Deletable attribute of these objects shall be ignored. Finally, the Program Invocation object shall be deleted. If any step of this process fails, the service shall fail, all partially completed steps shall be undone, and a Result(-) response shall be returned.

11.4 Start Service

The Start service allows an MMS client to change the state of a Program Invocation to the RUNNING state. The Program Invocation shall be in the IDLE state for this service to be successfully completed.

11.4.1 Structure

The structure of the component service primitives is shown in Table 30.

Table 30 – Start Service

Table 30 (Cont.) – Start Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Program Invocation Name	M	M(=)			
Execution Argument	U	U(=)			
Result(+) (COMP)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	
Program Invocation State			M	M(=)	

11.4.1.1 Argument

This parameter shall contain the parameters of the Start service request.

11.4.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be started.

11.4.1.1.2 Execution Argument

This parameter shall be an optional field which may be used to pass data to the started Program Invocation. This parameter shall be either a character string or an externally coded value.

11.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

11.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.4.1.3.1 Program Invocation State

This parameter shall be of type ProgramInvocationState. Following an unsuccessful Start service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the UNRUNNABLE state. This parameter shall identify the state of the Program Invocation following an unsuccessful Start.

11.4.2 Service Procedure

The MMS client shall identify the Program Invocation it wishes to start with the Program Invocation Name parameter. If the Program Invocation does not exist a Result(-) response shall be returned. The named Program Invocation shall be in the IDLE state or a Result(-) response shall be returned. If the Execution Argument parameter is present in the service indication, the Execution Argument attribute of the Program Invocation object shall have its value set to the value of the Execution Argument parameter; otherwise the value of the Execution Argument attribute shall remain unchanged. Upon receipt of the Start service indication primitive (after all pre-execution modifiers are satisfied) the Program Invocation shall be placed in the STARTING state. A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the RUNNING state. A Result(-) response primitive shall be issued if the starting process fails, and the Program

Invocation shall be returned either to the IDLE state if possible or to the UNRUNNABLE state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation. Since, in general, starting a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, then a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE – If the Program Invocation is implemented through a sequential procedural programming language, this service should cause execution to begin at the entry point of the program.

11.5 Stop Service

The Stop service allows an MMS client to cause a named Program Invocation on the MMS server to transition from the RUNNING state to the STOPPED state.

11.5.1 Structure

The structure of the component service primitives is shown in Table 31.

Table 31 – Stop Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Program Invocation Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	
Program Invocation State				M	M(=)	

11.5.1.1 Argument

This parameter shall contain the parameters of the Stop service request.

11.5.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation which is to be stopped.

11.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

11.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.5.1.3.1 Program Invocation State

This parameter, of type ProgramInvocationState, shall identify the final state of the Program Invocation if the Stop service fails.

11.5.2 Service Procedure

The MMS client shall identify the Program Invocation it wishes to stop with the Program Invocation Name parameter. If the Program Invocation does not exist a Result(-) response shall be returned. The named Program Invocation shall be in the RUNNING state or a Result(-) response shall be returned. If the Program Invocation exists and is in the RUNNING state, then upon receipt of the Stop service indication primitive (after all pre-execution modifiers are satisfied) the Program Invocation shall be placed in the STOPPING state. A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the STOPPED state. A Result(-) response primitive shall be issued if the stopping process fails, and the Program Invocation shall be returned either to the RUNNING state if possible or to the UNRUNNABLE state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation. Since, in general, stopping a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, then a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE – If the Program Invocation is implemented through a sequential procedural programming language, this service should cause the current program step to be saved as part of the control information of the Program Invocation for a subsequent resumption procedure.

11.6 Resume Service

The Resume service allows an MMS client to change the state of a Program Invocation to the RUNNING state. The Program Invocation shall be in the STOPPED state for this service to be successfully completed.

11.6.1 Structure

The structure of the component service primitives is shown in Table 32.

Table 32 – Resume Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Program Invocation Name		M	M(=)			
Execution Argument		U	U(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	
Program Invocation State				M	M(=)	

11.6.1.1 Argument

This parameter shall contain the parameters of the Resume service request.

11.6.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be resumed.

11.6.1.1.2 Execution Argument

This parameter shall be an optional field in which to pass data to the resuming Program Invocation. This parameter is either a character string or an externally coded value.

11.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

11.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.6.1.3.1 Program Invocation State

This parameter shall be of type ProgramInvocationState. Following an unsuccessful Resume service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the UNRUNNABLE state. This parameter shall identify the state of the Program Invocation following an unsuccessful Resume.

11.6.2 Service Procedure

The MMS client shall identify the Program Invocation it wishes to resume with the Program Invocation Name parameter. If the Program Invocation does not exist a Result(-) response shall be returned. The Program Invocation shall be in the STOPPED state or a Result(-) response shall be returned. If the Execution Argument parameter is present, the value of the Execution Argument attribute of the Program Invocation shall assume the value of the Execution Argument parameter. Otherwise, the value of the Execution Argument attribute shall remain unchanged. Upon receipt of the Resume service indication primitive (after all pre-execution modifiers are satisfied) the Program Invocation shall be placed in the RESUMING state. A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the RUNNING state. A Result(-) response primitive shall be issued if the resuming process fails, and the Program Invocation shall be returned either to the STOPPED state if possible, or to the UNRUNNABLE state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation. Since, in general, resuming a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, then a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE – If the Program Invocation is implemented through a sequential procedural programming language, this service should cause execution of the Program Invocation to continue at the program step indicated in the control information.

11.7 Reset Service

The Reset service allows an MMS client to cause a named Program Invocation at the MMS server to transition from the STOPPED state to the IDLE state or to the UNRUNNABLE state, according to the Reusable attribute value of the Program Invocation object.

11.7.1 Structure

The structure of the component service primitives is shown in Table 33.

Table 33 – Reset Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Program Invocation Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	
Program Invocation State				M	M(=)	

11.7.1.1 Argument

This parameter shall contain the parameters of the Reset service request.

11.7.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation which is to be reset.

11.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

11.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.7.1.3.1 Program Invocation State

This parameter, of type ProgramInvocationState, shall identify the final state of the Program Invocation if the Reset service fails.

11.7.2 Service Procedure

The MMS client shall identify the Program Invocation it wishes to reset with the Program Invocation Name parameter. If the Program Invocation does not exist a Result(-) response shall be be returned. The Program Invocation shall be in the STOPPED state or a Result(-) response shall be be returned. If the Program Invocation is in the STOPPED state then upon receipt of the Reset service indication primitive (after all pre-execution modifiers are satisfied) the Program Invocation shall be placed in the RESETTING state. A successful response primitive shall be issued as soon as the Program Invocation is placed in the IDLE state or the UNRUNNABLE state. A Result(-) response primitive shall be issued if the resetting process fails, and the Program Invocation shall be returned to the STOPPED state if possible, or to the UNRUNNABLE state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation. Since, in general, resetting a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, then a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE – If the Program Invocation is implemented through a sequential procedural programming language, this service should cause the control information of the Program Invocation to be altered to reflect that the Program Invocation is again at the "beginning of program".

11.8 Kill Service

The Kill service allows an MMS client to cause a named Program Invocation at the MMS server to be placed in the UNRUNNABLE state.

11.8.1 Structure

The structure of the component service primitives is shown in Table 34.

Table 34 – Kill Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+) (COMP)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.8.1.1 Argument

This parameter shall contain the parameter of the Kill service request.

11.8.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation which is to be killed.

11.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

11.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.8.2 Service Procedure

The MMS client shall identify the Program Invocation it wishes to kill with the Program Invocation Name parameter. If the Program Invocation does not exist a Result(-) response shall be returned. Upon receipt of the Kill indication primitive, the MMS server shall place the named Program Invocation in the UNRUNNABLE state. If the Kill service fails, the state of the Program Invocation shall be left unchanged.

11.9 GetProgramInvocationAttributes Service

The GetProgramInvocationAttributes service may be used to request that an MMS server return the attributes associated with the specified Program Invocation.

11.9.1 Structure

The structure of the component service primitives is shown in Table 35.

Table 35 – GetProgramInvocationAttributes Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M				
Program Invocation Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
State				M	M(=)	
List Of Domain Names				M	M(=)	
MMS Deletable				M	M(=)	
Reusable				M	M(=)	
Monitor				M	M(=)	
Execution Argument				M	M(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

11.9.1.1 Argument

This parameter shall contain the parameter of the GetProgramInvocationAttributes service request.

11.9.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the name of the Program Invocation whose attributes are requested.

11.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request has succeeded. When success is indicated, the following parameters shall be included.

11.9.1.2.1 State

This parameter, of type ProgramInvocationState, shall indicate whether the Program Invocation is in the IDLE, RUNNING, STOPPED, STARTING, STOPPING, RESUMING, RESETTING, or UNRUNNABLE states.

11.9.1.2.2 List Of Domain Names

This parameter, of type list of Identifier, shall provide the names of the Domains which are referenced by the specified Program Invocation.

11.9.1.2.3 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) this Program Invocation can be deleted using MMS services.

11.9.1.2.4 Reusable

This parameter, of type boolean, shall indicate whether this Program Invocation will return to IDLE state after execution (true) or will enter UNRUNNABLE state after execution (false).

11.9.1.2.5 Monitor

This parameter, of type boolean, shall specify, if true, that event monitoring is in force for this Program Invocation. If this parameter is true, the attributes of the related Event Enrollment object may be obtained using the GetEventEnrollmentAttributes service (15.14)

11.9.1.2.6 Execution Argument

This parameter shall contain the value of the Execution Argument attribute of the Program Invocation. This parameter shall be either a character string or an externally coded value.

11.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

11.9.2 Service Procedure

The MMS server shall verify that the specified Program Invocation exists. If the Program Invocation does not exist a Result(-) response shall be returned. Otherwise the attributes of the named Program Invocation shall be returned.

12 Variable Access Services

The variable access services provide facilities which allow a client MMS-user to access typed variables defined at the VMD. These facilities are provided by five objects in the VMD model, and by providing fourteen services which operate upon these objects. 12.1 describes the MMS model for variable access. This is followed in 12.2 by a description of MMS types, in 12.3 by a description of MMS-definable alternate access, in 12.4 by a description of MMS data values, and in 12.5 by a description of parameters which specify access to variables. The variable access services are then described in 12.6 through 12.19. Finally 12.20 describes the set of parameter conformance building blocks and static conformance requirements which apply for this clause.

CLARIFICATION – This clause describes the mapping between virtual objects, called MMS variables, and real objects, called "real" variables. MMS variables are not variables in the usual sense, but rather represent access paths to the underlying real objects. MMS variables do not have a "value" attribute, but through the V-Put and V-Get functions they provide an access method to the value of the underlying real variables. These real objects may be true system variables, or they may be system constants, that is configuration parameters, or they may represent system procedures which generate a value, or other objects. Real variables may have addresses and may be described as contiguous or noncontiguous. Such concepts do not apply to MMS variables. In this context the word "real" means "having concrete existence", not "floating-point".

12.1 The MMS Variable Access Model

A "variable" is an abstract element of the VMD which is capable of providing (when read) or accepting (when written), or both, a typed data value. A "type description" is an abstract description of the class of data which may be conveyed by a variable's value. A variable's type description determines its abstract syntax, its range of possible values, and its representation while being communicated using MMS.

The MMS variable access services have as their focal point the reading and writing of values of variables (or parts of variables) residing at the VMD. The following paragraphs describe the MMS variable access model by describing the variable access objects and by describing the services which create, manipulate, or destroy these objects.

MMS defines five variable access objects. They are

a) the Unnamed Variable object;

b) the Named Variable object;

c) the Scattered Access object;

d) the Named Variable List object;

e) the Named Type object.

A VMD which allows access to one or more of its real variables using the MMS variable access services shall provide a mapping between the real variables for which access is allowed and one or more MMS variable access objects.

NOTE – Prior to reading the remainder of the variable access model, the reader may find it helpful to review "Guidance To Implementors" in 12.21.

12.1.1 Objects Which Describe Variables

MMS defines two objects which describe the mapping between an MMS variable and a real variable at the VMD. They are the Unnamed Variable object and the Named Variable object. These objects provide for two distinct levels of abstraction.

The abstraction represented by the Unnamed Variable object is very close to the physical architecture of the real system. Unnamed Variable objects model device-dependent aspects of an addressable facility of the VMD. An Unnamed Variable object simply provides a mapping between the inherent type description represented by an address and the MMS representation for that type description. An Unnamed Variable object's existence corresponds with the existence of the VMD. This object is only available when the VADR parameter conformance building block is supported.

The abstraction represented by the Named Variable object models the application's view of a real variable at the VMD. A Named Variable object's existence may be specified to correspond with the life of a Domain, of an application association or of the VMD. The Named Variable object is only available when the VNAM parameter conformance building block is supported.

Though both Named Variable and Unnamed Variable objects may be used to model real variables at the VMD, there are significant differences between these objects and the facilities which they provide for the client MMS-user.

a) The Named Variable object describes access to the real variable using an application process determined name, whereas the Unnamed Variable object describes access to the real variable using a device-specific address.

b) The Named Variable object may be used to describe a variable which either does not have a known address (a computed variable) or which has an address that is not made public. The Unnamed Variable object requires a known, fixed address for the variable.

c) Finally, a Named Variable object may be used to describe a set of one or more application-related data elements which are accessed using a single name as a single operation (see 12.1.1.1), or to describe a more explicit type description for a set of one or more Unnamed Variable objects, or both. An Unnamed Variable object, on the other hand, models access to the built-in, implementation-dependent aspects of an addressable facility of the VMD.

MMS provides services which allow a client MMS-user to define Named Variable objects in terms of Unnamed Variable objects. These services are primarily intended for use in bridging the gap between the two environments for devices which do not directly support the VNAM-only (see 12.21.1) environment, whether due to age or simplicity, or both.

12.1.1.1 Requirements for Access to MMS Variables

The mapping of a Named Variable object or an Unnamed Variable object to a real variable shall ensure that access (to the real variable, using the object) either succeeds or fails, partial success shall not be reported.

NOTE – In the case of write failure, for example due to a hardware fault in the real system, a partial result from the write may be visible to a subsequent access, either local or remote.

Additionally, a VMD should guarantee, if possible, that access to an MMS variable is not interruptible. In other words, read access should return a value representing a single logical instant in the state of the VMD and write access should alter the state of the VMD at a single logical instant. Since such guarantees may not be possible, or if possible for some variables, may not be possible for all variables, the static conformance statement for the VMD shall state whether uninterruptible access is supported and, if supported, under what constraints it is guaranteed.

12.1.1.2 Relationship Between MMS Variables and the Real System

The relationship between a real variable and the MMS object which is used to access it is specified by a pair of abstract functions. These functions are described below.

12.1.1.2.1 The V-Get Function

The V-Get function obtains the value of an MMS variable from the state of the VMD. (Here, state encompasses all aspects of the VMDs operation which relate to the mapping of the MMS variable to the real system.) The parameters of the V-Get function are the state of the VMD, the variable's access method (including its address if the access method is PUBLIC), its type description and any alternate access which applies for the access.

If the access succeeds, the result is a value containing the accessed data elements of the real variable. The abstract type description of this value is specified by the type specification (12.2) and alternate access (12.3).

In the case of access failure, the result is a reason for failure. (See Data Access Error in 12.4.)

12.1.1.2.2 The V-Put Function

The V-Put function updates the state of the VMD with the value of an MMS variable. (Once again, state encompasses all aspects of the VMDs operation which relate to the mapping of the MMS variable to the real system). The parameters of the V-Put function are the variable's value, access method, type description and alternate access (if applicable). Given successful access, the MMS variable's type description and alternate access, if applicable, are applied to the MMS value to compute the real value of the variable and the result is used to update the state of the VMD. If this update fails, the resulting state of the VMD is unspecified and the access fails, giving a reason for failure.

12.1.1.3 Enforcement of Variable Access Privilege

Protection requirements (if any) for an MMS variable are inherited from the underlying real variable. These requirements are established by the VMD, based on local criteria. They are not specified by this part of ISO/IEC 9506. The value OBJECT-ACCESS-DENIED for the Data Access Error shall be used to indicate that a request for variable access has been denied due to insufficient privilege.

12.1.1.4 Unnamed Variable Object

The Unnamed Variable object describes the mapping of a single unnamed MMS variable to a real variable existing at a known and fixed address within the VMD. This mapping shall be such that the requirements of 12.1.1.1 are satisfied.

An Unnamed Variable object is never created or destroyed. Its existence is inherent in the architecture of the VMD. When VADR is supported, the content of every public (remotely visible) address is modeled as an Unnamed Variable object.

EXAMPLE – A general-purpose, byte-addressable device might assign the type "octet" to each public address. A special-purpose device with a small number of addresses might assign a specific type, based on the known use of the content of a specific address.

The attributes of the Unnamed Variable object are specified below, followed by a brief description of the services which operate on this object.

12.1.1.4.1 The Unnamed Variable Object - Attributes

Object: Unnamed Variable

```
        Key Attribute: Address
        Attribute: MMS Deletable (FALSE)
        Attribute: Access Method (PUBLIC)
        Attribute: Type Description
```

Address

The Address attribute shall provide the location of the real variable in the system which supports the VMD.

MMS Deletable

The MMS Deletable attribute for an Unnamed Variable object shall specify false.

Access Method

The Access Method attribute for an Unnamed Variable object shall specify PUBLIC access.

Type Description

The Type Description attribute shall indicate the inherent abstract type of the underlying real variable, when viewed using MMS services. It shall specify the class (bitstring, integer, floating-point, etc.) and range of values for the simple data elements of the variable, as well as the grouping of these data elements (into arrays or structures), if inherent in the address, for the purpose of access.

12.1.1.4.2 Operations On The Unnamed Variable Object

The services which operate upon the Unnamed Variable object are listed below.

a) Read - This service uses the V-Get function to obtain the current value of a real variable described by the Unnamed Variable object;

b) Write - This service uses the V-Put function to replace the current value of a real variable described by the Unnamed Variable object;

c) InformationReport - This service uses the V-Get function to obtain the current value of a real variable described by the Unnamed Variable object;

d) GetVariableAccessAttributes - This service returns the Type Description attribute of a Unnamed Variable object.

12.1.1.5 The Named Variable Object

The Named Variable object describes the mapping between an MMS variable and a real, application-defined, variable at the VMD. This mapping shall be such that the requirements of 12.1.1.1 are satisfied.

NOTE – It is recommended that a real variable have one and only one mapping to a Named Variable object. MMS does not enforce this recommendation.

The attributes of the Named Variable object are specified below, followed by a brief description of the services which operate on this object.

12.1.1.5.1 The Named Variable Object - Attributes

Object: Named Variable

```
Key Attribute: Variable Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Type Description
Attribute: Access Method (PUBLIC, ... )
Constraint: Access Method = PUBLIC
    Attribute: Address
```

Variable Name

The Variable Name attribute uniquely identifies a Named Variable object. A Variable Name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope. When a Named Variable object is temporarily created for use by a single access this attribute shall have the value UNDEFINED.

NOTE 1 – A Variable Name exists in the same name space as a Scattered Access Name.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) this Named Variable object may be deleted using the DeleteVariableAccess service.

Type Description

The Type Description attribute shall indicate the abstract type of the underlying real variable, when viewed using MMS services. It shall specify the class (integer, floating-point, etc.) and range of values for the simple data elements of the variable, as well as the grouping of these data elements (into arrays or structures), if applicable, for the purpose of access.

A variable's Type Description and Access Method shall provide sufficient information (to the VMD) to allow the value of each data element of the underlying real variable to be determined. A formal definition of MMS types is provided in the definition of the Type Specification parameter in 12.2.

Access Method

The Access Method attribute shall provide the information required by the VMD in order to locate the real variable. This attribute may contain any value which is appropriate to the VMD, including additional attributes which are not specified by this part of ISO/IEC 9506.

If the Access Method is declarable (and obtainable) using MMS services, then this attribute shall have the value PUBLIC, and the Address attribute shall be defined and available to client MMS-users requesting the attributes of the Named Variable object. Otherwise, the value of this attribute is a local issue. The PUBLIC access method shall not be available unless VADR is supported.

A VMD may declare an MMS variable which exists only at the instant of access. Such a variable does not have an address per se. It shall be defined by the VMD using local means and it shall have an Access Method other than PUBLIC. Also, a VMD may declare an MMS variable which does have an address, but choose not to reveal this address to client MMS-users. Such a variable shall also be locally defined with an Access Method other than PUBLIC.

Address

The Address attribute, which only exists for a Named Variable object having Access Method equal to PUBLIC, shall provide the location of the variable described by the Type Description attribute. The data elements of the variable, as described by the Type Description attribute, are located in contiguous addresses, starting with the first data element at the address specified by this attribute. The types of the data elements shall be compatible with the types of the data elements of the Unnamed Variable objects which are encompassed by these contiguous addresses. The determination of type "compatibility" is a local issue.

NOTE 2 – The intent is to allow the VMD to refuse to accept a definition containing a type conflict, for example a request to define a known boolean as a floating-point. The definition is left open in order to allow the VMD to accept a definition requesting a tightening of type specification, for example a declaration that a "word" (bitstring) contains an integer.

NOTE 3 – MMS does not provide for allocation of real variables in the VMD. MMS services may be used to describe where and how a real variable has been allocated. However, such usage requires detailed knowledge of the specific VMD implementation, including the specific application of the VMD in which the real variable is declared.

NOTE 4 – The requirement of contiguity of address is only for variables whose Access Method is PUBLIC.

12.1.1.5.2 Operations On The Named Variable Object

The services which operate upon the Named Variable object are listed below.

a) Read - This service uses the V-Get function to obtain the value of a real variable described by the Named Variable object;

b) Write - This service uses the V-Put function to replace the value of a real variable described by the Named Variable object;

c) InformationReport - This service uses the V-Get function to obtain the value of a real variable described by the Named Variable object;

d) DefineNamedVariable - This service creates a Named Variable object;

e) GetVariableAccessAttributes - This service returns the attributes of a Named Variable object;

f) DeleteVariableAccess - This service deletes a Named Variable object.

12.1.2 Objects Which Describe Access To Multiple Variables

MMS provides two objects which describe access to multiple MMS variables using a single name. They are the Scattered Access object and the Named Variable List object. These objects provide two distinct levels of service for the client MMS-user.

The Scattered Access object shall specify a mapping of a single MMS name to a constructed structure composed of independent MMS variables. Access to the underlying real variables using the Scattered Access object appears (to the client MMS-user) as though the Scattered Access object were mapped to a single real variable. Thus, access using a Scattered Access object is regulated by the requirements of 12.1.1.1. The Scattered Access object is only available when the VADR, VNAM, and VSCA parameter conformance building blocks are all supported.

The Named Variable List object shall specify a mapping of a single MMS name to a list of independent MMS variables. Unlike the Scattered Access object, however, the Named Variable List object is designed to maintain client awareness of the independence of the underlying real variables. This object provides a mechanism to avoid repeated transfer of frequently used variable access lists. Access using a Named Variable List succeeds or fails at the level of the members of the list. Therefore partial success may be reported.

12.1.2.1 The Scattered Access Object

The Scattered Access object provides for the assignment of a name for use when specifying access to a constructed "structure" composed of independent MMS variables. Each component of this structure shall be represented by either a Named Variable object, an Unnamed Variable object, or a Scattered Access object.

The requirements of 12.1.1.1 shall apply for access using this object. Thus, access using a Scattered Access object is analogous to accessing a single MMS variable having as components the variables referenced by the Scattered Access object.

NOTE – This object is primarily provided in order to support devices which, due to age or simplicity, do not provide for local definition of Named Variable objects having an access method allowing a variable's value to be determined apart from the specification of a single base address and which do not provide facilities for arranging the data elements of a variable so that they may be accessed using a single base address.

The attributes of the Scattered Access object are specified below, followed by a brief description of the services which operate on this object.

12.1.2.1.1 The Scattered Access Object - Attributes

Object: Scattered Access

```
Key Attribute: Scattered Access Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: List Of Component
    Attribute: Kind Of Reference (NAMED, UNNAMED, SCATTERED)
    Attribute: Reference
    Attribute: Component Name
    Attribute: Access Description
```

Scattered Access Name

The Scattered Access Name attribute shall uniquely identify a Scattered Access object. It is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope. When the Scattered Access object is temporarily created for the purpose of a single access, this attribute shall have the value UNDEFINED.

NOTE 1 – A Scattered Access Name exists in the same name space as a Variable Name. For most services (exceptions are DefineScatteredAccess and GetScatteredAccessAttributes) reference to a Scattered Access object is not distinguishable from reference to a Named Variable object.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) this Scattered Access object may be deleted using the DeleteVariableAccess service.

List Of Component

This attribute shall provide a list referencing one or more objects, each of which may be either a Named Variable object, an Unnamed Variable object, or a Scattered Access object. These objects describe the mapping for the component data elements of the Scattered Access object. The attributes of the elements of this list are described as follows.

Kind Of Reference

This attribute shall indicate the type of object referenced by the Reference attribute. Possible values are:

NAMED - the referenced object is a Named Variable object;

UNNAMED - the referenced object is an Unnamed Variable object;

SCATTERED - the referenced object is a Scattered Access object.

Reference

The Reference attribute shall contain a reference, of the type specified by the Kind Of Reference attribute, to an MMS object which either defines the mapping of an MMS variable to a real variable, or defines a scattered access mapping to a constructed structure of one or more independent MMS variables, each of which provides a mapping to a single real variable.

If Kind Of Reference equals NAMED or SCATTERED and the referenced object is deleted, then this attribute shall be set to UNDEFINED, resulting in an OBJECT-INVALIDATED error for the access specified by this Scattered Access object or any Scattered Access object to which this object is subordinate.

NOTE 2 – A referenced Unnamed Variable object is never invalidated or deleted.

Component Name

The Component Name attribute provides a name for use when specifying alternate access to this component of the Scattered Access object. It shall be either UNDEFINED, indicating that alternate access may not explicitly specify this component, or an Identifier, indicating that this component may be explicitly selected for access or for additional alternate access specification. (See 12.3.)

Access Description

The Access Description attribute provides a description of the desired access to the MMS variable (or Scattered Access object) represented by this component of the Scattered Access object. This attribute may have the value UNDEFINED, indicating that full access using the mapping defined by the referenced object is desired, or it may specify an alternate (possibly partial) access based upon the mapping defined by the referenced object.

When not equal to UNDEFINED, this attribute is used to alter the externally visible structuring of the data elements of the complex variable accessed using the mapping defined by the referenced object, or to specify access to some subset of the set of data elements which is included in the mapping defined by the referenced object, or both.

NOTE 3 – The Access Description does not alter the abstract type of an MMS variable's simple data elements.

The Access Description, along with the mapping defined by the referenced object shall provide sufficient information (to the VMD) to allow the value of each accessed data element of the referenced real variable to be determined. A formal definition of alternate access is provided in the definition of the Alternate Access parameter in 12.3.

12.1.2.1.2 Operations On The Scattered Access Object

The services which operate upon the Scattered Access object are listed below.

a) Read - This service uses the List Of Component attribute in order to determine the data elements (Named or Unnamed Variables) and performs the Read operation on these elements;

b) Write - This service uses the List Of Component attribute in order to determine the data elements (Named or Unnamed Variables) and performs the Write operation on these elements;

c) InformationReport - This service uses the List Of Component attribute in order to determine the data elements (Named or Unnamed Variables) and performs an Information Report service on these elements;

d) DefineScatteredAccess - This service creates a Scattered Access object;

e) GetVariableAccessAttributes - This service returns the externally visible type description of a Scattered Access object;

f) GetScatteredAccessAttributes - This service returns the actual attributes of a Scattered Access object;

g) DeleteVariableAccess - This service deletes a Scattered Access object.

12.1.2.2 The Named Variable List Object

The Named Variable List object specifies a name to be used for access to a list of independent MMS variables. Each element of this list shall be either a Named Variable object, an Unnamed Variable object, or a Scattered Access object. Access to the individual elements of the list is regulated by the requirements of 12.1.1.1. Access to the list as a whole is not. Thus, access using the Named Variable List object shall report success or failure for each object referenced by the list. Access using a Named Variable List object is analogous to independent accesses using the list's referenced variable access objects.

The attributes of the Named Variable List object are specified below, followed by a brief description of the services which operate on this object.

12.1.2.2.1 The Named Variable List Object - Attributes

Object: Named Variable List

```
Key Attribute: Variable List Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: List Of Variable
    Attribute: Kind Of Reference (NAMED, UNNAMED, SCATTERED)
    Attribute: Reference
    Attribute: Access Description
```

Variable List Name

The Variable List Name attribute shall uniquely identify a Named Variable List object. A Variable List Name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope. When a Named Variable List object is temporarily created for the purpose of a single access this attribute shall have the value UNDEFINED.

NOTE 1 – A Variable List Name exists in a separate name space from the Variable Name and Scattered Access Name.

MMS Deletable

The MMS Deletable attribute indicates whether (true) or not (false) this Named Variable List object may be deleted using the DeleteNamedVariableList service.

List Of Variable

This attribute shall provide a list referencing one or more objects, each of which may be either a Named Variable object, an Unnamed Variable object, or a Scattered Access object. The attributes of the elements of this list are described as follows.

Kind Of Reference

This attribute shall indicate the type of object referenced by the Reference attribute. Possible values are:

NAMED - the referenced object is a Named Variable object;

UNNAMED - the referenced object is an Unnamed Variable object;

SCATTERED - the referenced object is a Scattered Access object.

Reference

The Reference attribute shall contain a reference, of the type specified by the Kind Of Reference attribute, to an MMS object which either defines the mapping of an MMS variable to a real variable, or defines a scattered access mapping to a constructed structure of one or more independent MMS variables, each of which provides a mapping to a single real variable.

If Kind Of Reference equals NAMED and the referenced object is deleted, then this attribute shall be set to UNDEFINED, resulting in an OBJECT-INVALIDATED error for the access specified by this Named Variable object. If Kind Of Reference equals SCATTERED and the referenced object or any object which is subordinate to the referenced object is deleted, this attribute shall be set to UNDEFINED, resulting in an OBJECT-INVALIDATED error for the access specified by this Scattered Access object.

NOTE 2 – A referenced Unnamed Variable object is never invalidated or deleted.

Access Description

The Access Description attribute shall provide a description of the desired access to the variable (or variables) whose mapping is defined by the referenced Named Variable, Unnamed Variable, or Scattered Access object. This attribute may have the value UNDEFINED, indicating that full access using the mapping defined by the referenced object is desired, or it may specify an alternate (possibly partial) access based upon the mapping defined by the referenced object.

When not equal to UNDEFINED, this attribute is used to alter the externally visible structuring of the data elements of the complex variable accessed using the mapping defined by the referenced object, or to specify access to some subset of the set of data elements which are included in the mapping defined by the referenced object, or both.

NOTE 3 – Access Description does not alter the abstract type of an MMS variable's simple data elements.

The Access Description, along with the mapping defined by the referenced object shall provide sufficient information (to the VMD) to allow the value of each accessed data element of the referenced real variables to be determined. A formal definition of alternate access is provided in the definition of the Alternate Access parameter in 12.3.

12.1.2.2.2 Operations On The Named Variable List Object

The services which operate upon the Named Variable List object are listed below.

a) Read - This service uses the List Of Variable attribute in order to determine the variables which should be read and performs the Read service on these variables;

b) Write - This service uses the List Of Variable attribute in order to determine the variables which are to be written and performs the Write service on these variables;

c) InformationReport - This service uses the List Of Variable attribute in order to determine the variables to be reported and performs an Information Report service on these variables;

d) DefineNamedVariableList - This service creates a Named Variable List object;

e) GetNamedVariableListAttributes - This service returns the attributes of a Named Variable List object;

f) DeleteNamedVariableList - This service deletes a Named Variable List object.

12.1.3 The Named Type Object

The Named Type object shall provide for the assignment of a name to an MMS type description. This object is only available when both the VADR and the VNAM parameter conformance building blocks are supported.

The attributes of the Named Type object are specified below, followed by a brief description of the services which operate on this object.

12.1.3.1 The Named Type Object - Attributes

Object: Named Type

```
Key Attribute: Type Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Type Description
```

Type Name

The Type Name attribute shall uniquely identify a Named Type object. A Type Name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) this Named Type object may be deleted using the DeleteNamedType service.

Type Description

The Type Description attribute shall describe an abstract MMS type which may be applied to a real variable in order to allow its value to be communicated using MMS.

12.1.3.2 Operations On The Named Type Object

The services which operate upon the Named Type object are listed below.

a) DefineNamedType - This service creates a Named Type object;

b) GetNamedTypeAttributes - This service returns the attributes of a Named Type object;

c) DeleteNamedType - This service deletes a Named Type object;

d) Read, Write, DefineNamedVariable, DefineScatteredAccess, DefineNamedVariableList, DefineNamedType - These services use the Type Description attribute in order to resolve a Type Name parameter included in the service request.

12.2 Specification of Types

All MMS variables are typed. A variable's type description, as contained in the Type Description attribute of the Named Variable or Unnamed Variable object, or as specified by the Variable Description parameter's Type Specification parameter at the time of access, shall provide a specification of the abstract syntax and range of possible values of the variable, and also provide the basis upon which alternate access to the variable may be specified.

The type description of a variable may be simple, specifying access to a single data element, or complex, specifying access to a group of related simple types. For each complex variable (array or structure), from the information available in a variable's access method (and address, if applicable), and its type description, the VMD shall be able to locate every simple data element of the real variable.

12.2.1 Type Specification Parameter

In the various variable access services, the type description of a variable, or the definition of a Type Name, shall be specified by the Type Specification parameter. This parameter itself describes a procedure by which a Type Description attribute is generated. This generation shall always be performed at the time of generation of the MMS variable object with the effect that all references to Type Name parameters are resolved at the time of definition, and Type Descriptions never explicitly depend on Named Type objects.

NOTE – As a consequence of this immediate evaluation rule, it is possible to define a VMD-specific variable in terms of a AA-specific Named Type. For example, a variable will contain a correct Type Description attribute even if a referenced AA-specific Named Type object is deleted because the application association disappears.

The structure of this parameter is given in Table 36.

Table 36 – Type Specification Parameter

Table 36 (Cont.) – Type Specification Parameter

Parameter Name	Req Rsp	Ind Cnf	CBB
Kind Of Type	M	M(=)	
Type Name	S	S(=)	
Array	S	S(=)	STR1
Packed	M	M(=)	VADR
Number of Elements	M	M(=)	
Type Specification	M	M(=)	
Structure	S	S(=)	STR2
Packed	M	M(=)	VADR
List Of Components	M	M(=)	
Component Name	U	U(=)	VALT
Type Specification	M	M(=)	
Simple	S	S(=)	
Class	M	M(=)	
Size	C	C(=)	

As can be seen from Table 36, a type is described by a recursively specified parameter. This parameter describes a branching tree, called a type tree. The leaves of this tree are the simple data elements of the variable described by the type. If a type describes a complex variable (an array or a structure), then the type tree will have one or more non-leaf nodes, each of which represents a complex type constructed from the (possibly complex) types represented by the subordinate nodes of the type tree.

The parameters of the Type Specification parameter are as follows.

12.2.1.1 Kind Of Type

The Kind Of Type parameter shall indicate the selection which has been chosen to describe this node of the type tree. The values for this parameter are:

TYPE-NAME - shall indicate that the Type Name parameter is selected. This value shall not occur in a response or confirm service primitive, or in an InformationReport.indication primitive.

ARRAY - shall indicate that the Array parameter is selected.

STRUCTURE - shall indicate that the Structure parameter is selected.

SIMPLE - shall indicate that the Simple parameter is selected.

12.2.1.2 Type Name

The Type Name selection for the Type Specification parameter shall indicate that this node of the type tree is to inherit its definition from the Type Description attribute of the Named Type object having a Type Name attribute equal to this parameter. (The Type Name is replaced by the value of the Named Type object's Type Description attribute).

12.2.1.3 Array

This selection for the Type Specification parameter shall indicate that the node being described is a complex type that is constructed from an ordered sequence of elements of a single type, with elements numbered from zero, the first element, and increasing.

NOTE – From a modelling perspective, an array is described by the number of sub-trees specified by the Number Of Elements parameter, each having the type specified by the Type Specification parameter, and packing, as specified by the Packed parameter, and each immediately subordinate to the node of the type tree which specifies the array type. A given array element is identified by the position of its sub-tree under the array's type tree. The first array element is identified by zero, the second by one, and so forth, with the last element identified by the value of Number Of Elements minus one.

12.2.1.3.1 Packed

This parameter shall specify whether or not storage optimization rules are in effect for locating the data elements of this array (and its subordinate types). When false, unless defined subordinate to a type for which this attribute is true, storage optimization rules are not in effect. When true, storage optimization rules are in effect for the entire sub-tree described by the array type.

For Type Specification parameters which specify the type of a Named Variable object having Access Method equal to PUBLIC or which specify the type associated with an Unnamed Variable object, this parameter may be true or false, as applicable to the variable. Otherwise, this parameter shall be false.

NOTE – The specific rules for locating the data elements of a storage optimized array are determined by the implementation. These rules should ensure the integrity of surrounding data elements in the case of partial access, if allowed.

12.2.1.3.2 Number Of Elements

This parameter shall specify the number of elements of the array.

12.2.1.3.3 Type Specification

This parameter shall specify the type description of the array elements through a recursive reference to the Type Specification parameter.

12.2.1.4 Structure

The Structure parameter shall specify that this node of the type tree describes a complex type that is constructed from an ordered list of one or more components, each of which may have a distinct type.

12.2.1.4.1 Packed

This parameter shall specify whether storage optimization rules are in effect for locating the data elements of this structure (and its subordinate types) or not. When false, unless defined subordinate to a type for which this attribute is true, storage optimization rules are not in effect. When true, storage optimization rules are in effect for the entire sub-tree described by the structure type.

For Type Specification parameters which specify the type of a Named Variable object having Access Method equal to PUBLIC or which specify the type associated with an Unnamed Variable object, this parameter may be true or false, as applicable to the variable. Otherwise, this parameter shall be false.

NOTE – The specific rules for locating the data elements of a storage optimized structure are locally determined. These rules should ensure the integrity of surrounding data elements in the case of alternate access, if allowed.

12.2.1.4.2 List Of Components

The List Of Components parameter shall describe the components of the structure. At least one component shall be described.

12.2.1.4.2.1 Component Name

The Component Name parameter, of type Identifier, shall uniquely identify the component within the scope of the structure (node) to which the component is an immediate subordinate. This parameter is required if this node may be referenced by an alternate access specification. Otherwise, this parameter may be omitted.

12.2.1.4.2.2 Type Specification

This parameter shall specify the type description of the structure component through a recursive reference to the Type Specification parameter.

12.2.1.5 Simple

The Simple selection for the Type Specification parameter shall indicate that a leaf node of the type tree is being described. Such a node shall contain the class of the data element represented by the node and, when applicable, the size (or precision) associated with the specific instance of the class. The class and size together serve to specify the range of possible values for the variable.

12.2.1.5.1 Class

The Class parameter of the Simple selection shall specify the class of the data element represented by the leaf node. The value of this parameter shall be chosen from one of the following values.

a) BOOLEAN - The definition of this MMS type is as specified for the boolean type in ISO 8824. The Size parameter shall be omitted.

b) BIT STRING - The definition of this type is as specified for the bitstring type in ISO 8824. The Size parameter shall specify the number of bits in the bit string and an indication of whether this is an absolute number (indicating a fixed-length bitstring) or a maximum number (indicating a variable-length bitstring).

c) INTEGER - The definition of this type is as specified for the integer type in ISO 8824. The Size parameter shall specify the number of bits (assuming twos-complement representation) required in order to allow representation of all possible distinguished values.

d) UNSIGNED - The definition of this type is as specified for the integer type in ISO 8824, with the exclusion of the negative whole numbers. The Size parameter shall contain the number of bits (assuming binary representation) required in order to allow representation of all possible distinguished values.

e) FLOATING POINT - This class defines a simple type with distinguished values which are the positive and negative real numbers, including zero, and including a representation for positive and negative infinity, and NaN (not a number). The Size parameter shall specify in bits the format width, F, and the exponent width, E. The format width is the number of bits used to represent the floating point value including sign, exponent, and fraction.

   NOTE — The terms "positive infinity", "negative infinity", "NaN", "format width" and "exponent width" are defined in the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Standard 754 - 1985). MMS allows any number of bits in the format and exponent. This includes, but is not limited to, the two basic formats defined in that standard. Additional information may be found in ISO/IEC 9506-2.

f) REAL - The definition of this type is as specified for the real type in ISO 8824. The Size parameter shall contain three numbers which represent the base of the representation which shall be two or ten, the maximum number of octets used to represent the exponent, and the maximum number of octets used to represent the mantissa. The second and third values are meaningful only if the base of the representation is two.

g) OCTET STRING - The definition of this type is as specified for the octetstring type in ISO 8824. The Size parameter shall contain the number of octets in the octetstring and an indication of whether this is an absolute number (indicating a fixed-length string) or a maximum number (indicating a variable-length string).

h) VISIBLE STRING - The definition of this type is as specified for the VisibleString type in ISO 8824. The Size parameter shall contain the number of characters in the string and an indication of whether this is an absolute number (indicating a fixed-length string) or a maximum number (indicating a variable-length string).

i) GENERALIZED TIME - The definition of this type is as specified for the GeneralizedTime type in ISO 8824. The Size parameter shall be omitted.

j) BINARY TIME - The definition of this type is as specified for the TimeOfDay type in this part of ISO/IEC 9506. The Size parameter shall indicate whether (true) or not (false) the date is to be included in values of the type.

k) BCD - This type shall consist of the set of distinguished values which are sequences of one or more numeric digits (0, 1, ... 9) from some character set. The Size parameter shall indicate the absolute number of digits of the type.

l) OBJECT IDENTIFIER - The definition of this type is as specified for the object identifier type in ISO 8824. The Size parameter shall be omitted.

12.2.1.5.2 Size

The presence (or not) of the Size parameter, as well as its meaning, is dependent upon the value of the Class parameter, as specified above. When Class specifies a variable length string type, Size shall be an integer determined by subtracting the maximum length of the string from zero.

12.3 Specification of Alternate Access

As described in 12.2, all MMS variables are typed. A variable's type describes the abstract syntax and range of possible values of the real variable in the VMD. An alternate access description specifies an alternate view of this type. It may be used to alter the perceived abstract syntax of the variable (as seen using MMS services) or to restrict access to a subset of the range of possible values of the variable (partial access) or both.

Alternate access to a variable (or Scattered Access object), as contained in the Access Description attribute of the Named Variable List or Scattered Access object or as provided by the Alternate Access parameter of a specific access request, provides a mapping from (to) the view provided by the referenced MMS object to (from) the view which is desired by the access. This results in an indirect mapping to the real variable.

In the various variable access services, alternate access is represented by the presence of the Alternate Access parameter. The definition for this parameter fully defines the Access Description attribute (when not UNDEFINED) of the Named Variable List and Scattered Access objects. It also describes the derivation of the "derived type" which results from applying an alternate access to a type, whether derived or explicitly specified by a type specification. This derived type determines the abstract syntax of the Data parameter (12.4), when used to convey values associated with the alternate access. It also determines the type tree which is used when describing alternate access to the mapping provided by a Scattered Access object which itself specifies alternate access to some subordinate variable access object.

The description of the Alternate Access parameter is based upon its relationship to an MMS variable's type. Following this description, the relationship between the alternate access parameter and the mapping provided by a Scattered Access object will be described.

12.3.1 Alternate Access Parameter

The structure of the Alternate Access parameter is given in Table 37.

Table 37 – Alternate Access Parameter

Conformance: VALT Parameter Name	Req Rsp	Ind Cnf	CBB
List Of Alternate Access Selection	M	M(=)	
Component Name	U	U(=)	
Kind Of Selection	M	M(=)	
Select Alternate Access	S	S(=)	
Access Selection	M	M(=)	
Component	S	S(=)	STR2
Index	S	S(=)	STR1
Index Range	S	S(=)	STR1
Low Index	M	M(=)	
Number Of Elements	M	M(=)	
Alternate Access	M	M(=)	
Select Access	S	S(=)	
Access Selection	M	M(=)	
Component	S	S(=)	STR2
Index	S	S(=)	STR1
Index Range	S	S(=)	STR1
Low Index	M	M(=)	
Number Of Elements	M	M(=)	

12.3.1.1 List Of Alternate Access Selection

This parameter shall specify a list containing one or more Alternate Access Selection parameters. Each Alternate Access Selection parameter selects a node or, in the case of an array, a range of nodes, at the next higher nesting level of the type tree. This selection may be for the purpose of additional alternate access specification or for specifying access to the data elements represented by the selected nodes.

When the List Of Alternate Access Selection parameter contains more than one element, the derived type resulting from this parameter is a structure. The components of this derived type have component names and types as determined by the selections specified in the list.

When the List Of Alternate Access Selection parameter contains exactly one element, the derived type resulting from this parameter is determined by the selection specified for it, and the Component Name parameter shall not be specified.

12.3.1.1.1 Component Name

This parameter shall not be present if the List Of Alternate Access Selection specifies a single selection. Otherwise it is optional, and if present, shall specify the name of this component of the alternate access for use in specifying alternate access to the type derived from this application of the Alternate Access parameter.

12.3.1.1.2 Kind Of Selection

This parameter shall indicate whether access or further recursion in the alternate access specification is being specified. The possible values are:

SELECT-ALTERNATE-ACCESS - Indicates that further recursion in the alternate access specification is being specified and that the Select Alternate Access parameter is present.

SELECT-ACCESS - Indicates that access (read or write) is being specified and that the Select Access parameter is present.

12.3.1.1.3 Select Alternate Access

This parameter is selected for the Alternate Access Selection when one or more sub-trees at the next higher nesting level of the type tree are to be selected for further (recursive) alternate access specification. If the current node is an array, it shall select a single array element, a sub-range of array elements, or all array elements for further alternate access specification. If the current node is a structure, it shall select a single component of the structure for further alternate access specification. The parameters of Select Alternate Access are as follows.

12.3.1.1.3.1 Access Selection

This parameter shall indicate which of the selections for specifying alternate access has been taken. The possible values are:

COMPONENT - selects a single component of the structure as identified by the Component parameter, for alternate access.

INDEX - selects a single array element, as specified by the Index parameter, for alternate access.

INDEX-RANGE - selects an array of elements, as specified by the Index Range parameter, for alternate access.

12.3.1.1.3.2 Component

The Component parameter, of type Identifier, shall be selected if the current node of the type tree specifies a structure and the Access Selection parameter indicates COMPONENT. This parameter selects, for further alternate access specification, the specific structure component having Component Name equal to the Component parameter. The selected component shall be an array or a structure. The type tree node representing the selected structure component's type description is selected and the Alternate Access parameter is applied to that node. The derived type which results from applying the Component selection is determined by application of the Alternate Access parameter to the selected type tree node.

12.3.1.1.3.3 Index

The Index parameter, of type integer, shall be selected if the current node of the type tree specifies an array and the Access Selection indicates INDEX. Otherwise this parameter shall not be selected. It selects the specific array element (which shall be an array or a structure) for further alternate access specification. The type tree node representing the selected array element's type description is selected and then the Alternate Access parameter is applied to this node. The derived type which results from applying the Index selection is determined by application of the Alternate Access parameter to the selected node.

12.3.1.1.3.4 Index Range

The Index Range parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX-RANGE. It selects a range of array elements (having array or structure type) for further alternate access specification. For each element of the selected range, in order of increasing index, the type tree node representing that element's type description is selected and the Alternate Access parameter is applied to that node. The derived type which results from applying the Index Range selection is an array having elements of the type determined by application of the Alternate Access parameter to each of the selected nodes. (These elements are numbered from zero as for any other array).

12.3.1.1.3.4.1 Low Index

This integer parameter shall indicate the start of the index range. It shall be a valid index of the array. The specified element shall be the first element of the resulting derived array type and shall be numbered zero in that type.

12.3.1.1.3.4.2 Number Of Elements

This integer parameter shall indicate the number of elements to be included in the index range, including the element selected by Low Index. If this parameter has the value zero, then all elements having index greater than or equal to Low Index and less than or equal to the maximum index for the array are selected. If greater than zero, then all elements having index between Low Index and Low Index plus Number Of Elements minus one, inclusive are selected. Each of the selected elements shall be defined.

12.3.1.1.3.5 Alternate Access

The Alternate Access parameter shall specify additional alternate access at the selected node (or nodes).

12.3.1.1.4 Select Access

This parameter is selected for the Alternate Access Selection when one or more sub-trees at the next higher nesting level of the type tree are desired for access (read or write). If the current node is an array, it shall select a single array element or a sub-range of array elements for access. If the current node is a structure, it shall select a single component of the structure for access. The entire sub-tree (including all of its data elements) specified by the selected node (or nodes) is accessed. The parameters of Select Access are as follows.

12.3.1.1.4.1 Access Selection

This parameter shall indicate which of the selections for specifying access has been taken. The possible values are:

COMPONENT - selects a single component of the structure as identified by the Component parameter, for access.

INDEX - selects a single array element, as specified by the Index parameter, for access.

INDEX-RANGE - selects an array of elements, as specified by the Index Range parameter, for access.

12.3.1.1.4.2 Component

The Component parameter shall be selected if the current node of the type tree specifies a structure and the Access Selection parameter indicates COMPONENT. Otherwise, this parameter shall not be selected. It selects a specific structure component for access. The derived type resulting from the selection is the type of the selected structure component.

12.3.1.1.4.3 Index

The Index parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX. Otherwise, this parameter shall not be selected. It selects the specific array element to be accessed. The derived type resulting from the selection is the type of the selected array element.

12.3.1.1.4.4 Index Range

The Index Range parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX-RANGE. Otherwise, this parameter shall not be selected. It selects a range of array elements which are to be accessed. The derived type resulting from the Index Range selection is an array containing the selected elements, each having the type of the array element. (These elements are numbered from zero as for any other array).

12.3.1.1.4.4.1 Low Index

This integer parameter shall indicate the start of the index range. It shall be a valid index of the array. The specified element shall be the first element of the resulting derived array type and shall be numbered zero in that type.

12.3.1.1.4.4.2 Number Of Elements

This integer parameter shall indicate the number of elements to be included in the index range, including the element selected by Low Index. If this parameter has the value zero, then all elements having index greater than or equal to Low Index and less than or equal to the maximum index for the array are selected. If greater than zero, then all elements having index between Low Index and Low Index plus Number Of Elements minus one, inclusive are selected. Each of the selected elements shall be defined.

12.3.2 Application of Alternate Access to Scattered Access Objects

The previous discussion described the effect of applying an alternate access specification to a variable's type description. As indicated in this discussion, application of alternate access to a type description results in selection of the specific nodes (data elements) of the type tree which are to be accessed. These data elements are then represented to an MMS client using a "derived" type. This derived type controls the specification of the Data parameter when values of the alternately accessed variable are communicated.

When the Scattered Access object is supported by a VMD, it is possible to specify alternate access to a Scattered Access object. The alternate access applies to the derived type resulting from the Scattered Access object's definition, which may include components specified using alternate access.

The derived type of a Scattered Access object is always a structure. This structure has as its components the MMS variables or Scattered Access objects which are listed in the Scattered Access object's List Of Component attribute. Each of these components, in turn, has a derived type which results from applying the component's alternate access attribute to the type (derived or explicit) of the referenced object.

When alternate access is specified for a Scattered Access object, the Alternate Access parameter selects among the nodes of the type tree of the derived type represented by the Scattered Access object. The Component Name attribute of each of the components listed in the Scattered Access object's List Of Component attribute, and the component names and index ranges which were specified (as parameters of the Alternate Access parameter) when the Scattered Access object's components were specified, form the basis for selections which may be made when alternate access is requested for the Scattered Access object.

The result of applying alternate access to a Scattered Access object is a derived type. This derived type controls the specification of the Data parameter when values of the alternately accessed Scattered Access object are communicated.

12.4 Specification of Data Values

The MMS Read, Write and InformationReport services are used to convey the desired (Write) or current (Read and InformationReport) value of an MMS variable or a set of MMS variables referenced by a Scattered Access object.

For Read and InformationReport, the current value of an MMS variable is obtained by applying the V-Get function (see 12.1.1.2), and the result is represented by the Access Result parameter. The current value of a constructed "variable" represented by a Scattered Access object is obtained by applying the V-Get function to the MMS variables of the Scattered Access object. If all such applications result in success, the result of the access is transferred using the derived type represented by the Scattered Access object's mapping, otherwise the reason for failure is returned.

For Write, the desired value of a variable is represented by the Data parameter. For an MMS variable, this value updates the current state of the VMD by application of the V-Put function (see 12.1.1.2). The result of this update, if successful, is represented by a simple confirmation. If the update fails, the result is represented by the Data Access Error parameter. For a Scattered Access object, this value updates the current state of the VMD by application of the V-Put function (see 12.1.1.2) for each MMS variable represented by the Scattered Access mapping. The result of this update, if successful, is represented by a simple confirmation. If any portion of this update fails, the result is represented by the Data Access Error parameter.

The Access Result, Data and Data Access Error parameters are specified below.

12.4.1 Access Result Parameter

The Access Result parameter is used by the Read and InformationReport services in order to inform the client of the result of reading an MMS variable or a constructed "variable", as specified by a Scattered Access object.

The structure of this parameter is given in Table 38.

Table 38 – Access Result Parameter

Parameter Name	Req Rsp	Ind Cnf	CBB
Success	M	M(=)	
Data Access Error	S	S(=)	
Data	S	S(=)	

12.4.1.1 Success

This parameter shall indicate whether (true) or not (false) the access succeeded, and shall specify the selection for the following parameters.

12.4.1.2 Data Access Error

If the Success parameter indicates that the access failed, this parameter shall contain the reason for failure. This parameter is described in 12.4.3.

12.4.1.3 Data

If the Success parameter indicates that the access succeeded, this parameter shall contain the value of the variable (or constructed variable). The abstract syntax of a Data value shall be determined by the derived type specified by the access (alternate access or scattered access) or by the variable's defined type (Named or Unnamed Variable object).

The parameters of the Data parameter are described in 12.4.2.

12.4.2 Data Parameter

The Data parameter is used by the Read, Write and InformationReport services to convey the value of a variable or a constructed "variable" described by a Scattered Access object.

The structure of the component parameters is shown in Table 39.

Table 39 – Data Parameter

Table 39 (Cont.) – Data Parameter

Parameter Name	Req Rsp	Ind Cnf	CBB
Kind Of Data	M	M(=)	
Array	S	S(=)	STR1
List Of Data	M	M(=)	
Structure	S	S(=)	STR2
List Of Data	M	M(=)	
Simple	S	S(=)	
Class	M	M(=)	
Value	M	M(=)	

As can be seen from Table 39, this parameter is recursively defined. It shall specify a branching tree. Each node of this tree corresponds to a node of the variable's type tree, or derived type tree after applying scattered or alternate access (or both), see 12.2 and 12.3.

12.4.2.1 Kind Of Data

The value of the Kind Of Data parameter shall be ARRAY, STRUCTURE or SIMPLE depending upon whether the value at the current node of the type tree is an array, a structure, or a simple data element, respectively.

12.4.2.2 Array

The Array parameter shall be selected when Kind Of Data indicates an array. It shall specify an ordered list of Data parameters. Each element of this list provides the value of the corresponding element of the array. The elements of the list are ordered from array element zero and increasing to the last array element.

12.4.2.3 Structure

The Structure parameter shall specify an ordered list of Data parameters. Each element of this list shall specify the value of the corresponding component of the structure. The elements of the list are ordered from the first component to the last component of the structure.

12.4.2.4 Simple

The Simple parameter shall be selected when Kind Of Data indicates a simple data element. It shall specify the class and value of a simple data element of the variable.

12.4.2.4.1 Class

This parameter shall indicate the class of data conveyed by the current value. Its possible values are BOOLEAN, BIT STRING, INTEGER, UNSIGNED, FLOATING POINT, REAL, OCTET STRING, VISIBLE STRING, GENERALIZED TIME, BINARY TIME, BCD, and OBJECT IDENTIFIER as specified for the Simple type of the Type Specification parameter, in 12.2.1.5.1.

12.4.2.4.2 Value

This parameter shall contain the actual value of the simple data element.

12.4.3 Data Access Error Parameter

The Data Access Error parameter shall indicate the reason for failure of an attempted access to a variable or a set of variables specified by a Scattered Access object. The possible values for this parameter are as follows.

OBJECT-INVALIDATED - An attempted access references a defined object which has an undefined reference attribute. This represents a permanent error for access attempts to that object.

HARDWARE-FAULT - An attempt to access the variable has failed due to a hardware fault.

TEMPORARILY-UNAVAILABLE - The requested variable is temporarily unavailable for the requested access.

EXAMPLE – A VMD may disallow an attempt to write to the set-point of a control loop which has been placed in manual mode.

OBJECT-ACCESS-DENIED - The MMS client has insufficient privilege to request this operation.

OBJECT-UNDEFINED - The object with the desired name does not exist.

INVALID-ADDRESS - Reference to the unnamed variable object's specified address is invalid because the specified format is incorrect or is out of range.

TYPE-UNSUPPORTED - An inappropriate or unsupported type is specified for a variable.

TYPE-INCONSISTENT - A type is specified which is inconsistent with the service or referenced object.

OBJECT-ATTRIBUTE-INCONSISTENT - The object is specified with inconsistent attributes.

OBJECT-ACCESS-UNSUPPORTED - The variable is not defined to allow requested access.

OBJECT-NON-EXISTENT - The variable is nonexistent.

NOTE – The Data Access Error parameter does not indicate failure of a service request. Instead, it indicates failure of an attempted valid access request. The OBJECT-INVALIDATED error indicates that the mapping represented by a reference contained in a defined Named Variable List or Scattered Access object is no longer valid. All other values indicate failure of the V-Get or V-Put function.

12.5 Specification of Access to Variables

This subclause describes the parameters which specify access to a variable. These include the Variable Access Specification Parameter, the Variable Specification Parameter, the Scattered Access Description Parameter, and the Address parameter.

12.5.1 Variable Access Specification Parameter

The structure of the Variable Access Specification parameter is shown in Table 40.

Table 40 – Variable Access Specification Parameter

Parameter Name	Req Rsp	Ind Cnf	CBB
Kind Of Access	M	M(=)	
List Of Variable	S	S(=)	
Variable Specification	M	M(=)	
Alternate Access	U	U(=)	VALT
Variable List Name	S	S(=)	VLIS

12.5.1.1 Kind Of Access

This parameter shall indicate whether the access is specified in terms of an enumerated list of Variable Specification and (optionally) Alternate Access parameters or in terms of a single Variable List Name parameter giving the value of the Variable List Name attribute of a Named Variable List object defined at the VMD.

12.5.1.2 List Of Variable

When Kind Of Access specifies an enumerated list, this parameter shall be specified, otherwise it shall not be specified. When specified, this parameter shall list each variable (one or more) to be accessed, along with any Alternate Access which shall apply. Each element of this list shall contain the following parameters.

12.5.1.2.1 Variable Specification

The Variable Specification parameter shall identify the variable (at the VMD) whose value:

a) is to be read (Read request and indication service primitives);

b) is to be written (Write service); or

c) has been read (InformationReport service and Read response and confirm service primitives).

The parameters of the Variable Specification parameter are specified in 12.5.2.

12.5.1.2.2 Alternate Access

This parameter shall specify the alternate access which is applicable for this service instance. If not included, full access as specified by the variable's definition is to be used.

This parameter shall be omitted if the variable's type is simple.

12.5.1.3 Variable List Name

When Kind Of Access specifies a named list, this parameter, of type Object Name, shall be specified, otherwise it shall not be specified. When specified this parameter shall provide the value of the Variable List Name attribute of a Named Variable List object at the VMD. This object shall specify a list of one or more variables which:

a) are to be read (Read request and indication service primitives);

b) are to be written (Write service); or

c) have been read (Read response and confirm service primitives and InformationReport service).

12.5.2 Variable Specification Parameter

The Variable Specification parameter shall specify access to a single MMS variable or constructed "variable" represented by a Scattered Access object. The structure of its component parameters is shown in Table 41.

Table 41 – Variable Specification Parameter

Table 41 (Cont.) – Variable Specification Parameter

Parameter Name	Req Rsp	Ind Cnf	CBB
Kind Of Variable	M	M(=)	
Name	S	S(=)	VNAM
Address	S	S(=)	VADR
Variable Description	S	S(=)	VADR
Address	M	M(=)	
Type Specification	M	M(=)	
Scattered Access Description	S	S(=)	VSCA

12.5.2.1 Kind Of Variable

This parameter shall indicate the kind of variable access which is to be (or has been) performed. Its value shall be selected from the following:

NAMED - indicating access using a Named Variable or Scattered Access object. When this value is selected the Name Parameter shall be present.

UNNAMED - indicating access using an Unnamed Variable object. When this value is specified the Address parameter shall be present.

SINGLE - indicating access using a temporarily created Named Variable object whose definition is supplied in the access request. When this value is specified the Variable Description parameter shall be present.

SCATTERED - indicating access using a temporarily created Scattered Access object whose definition is supplied in the access request. When this value is specified, the Scattered Access Description parameter shall be present.

INVALIDATED - indicating an attempted access to an invalidated variable. This value may only occur in a response or a confirm primitive. When this value is specified, the Name, Address, Variable Description, and Scattered Access Description parameters shall be absent.

12.5.2.2 Name

The Name parameter, of type Object Name, shall specify the Variable Name attribute of a Named Variable object or the Scattered Access Name of a Scattered Access object.

12.5.2.3 Address

The Address parameter shall specify access using the built-in type specified by an Unnamed Variable object. The parameters of the Address parameter are given in 12.5.4.

12.5.2.4 Variable Description

The Variable Description parameter shall specify Address and Type Specification for access using a temporarily created Named Variable object. The created object is deleted following the access.

12.5.2.4.1 Address

The Address parameter shall specify the address of the variable being described. It represents the base address of the variable, as described by the Type Specification parameter.

The Address parameter is described in 12.5.4.

12.5.2.4.2 Type Specification

The Type Specification parameter shall specify the variable's abstract type. The specified type shall be compatible with the Type Description attributes of all Unnamed Variable objects which are included in the variable. The definition of "compatible" is a local issue. (See description of the Address attribute in 12.1.1.5.1). The Type Specification parameter is described in 12.2.

12.5.2.5 Scattered Access Description

The Scattered Access Description parameter shall specify access using a Scattered Access object. The Scattered Access Description parameter is described below.

12.5.3 Scattered Access Description Parameter

The Scattered Access Description parameter is used to describe an access to a "structure" composed of one or more independent MMS variables. The structure of this parameter is given in Table 42.

Table 42 – Scattered Access Description Parameter

Conformance: VSCA,STR2 Parameter Name	Req Rsp	Ind Cnf	CBB
List Of Component	M	M(=)	
Component Name	U	U(=)	VALT
Variable Specification	M	M(=)	
Alternate Access	U	U(=)	VALT

12.5.3.1 List Of Component

This parameter shall contain an ordered list specifying one or more Named Variable, Unnamed Variable or Scattered Access objects, in any combination, which are to comprise the components of this scattered access.

12.5.3.1.1 Component Name

This parameter, of type Identifier, is optional. If omitted, this scattered access component may not be referenced by an alternate access. If included, this parameter shall specify the name which shall apply for this component should alternate access be requested.

12.5.3.1.2 Variable Specification

This parameter shall specify the Named Variable, Unnamed Variable, or Scattered Access object which defines the location and type of the data element(s) of this component of the scattered access, as well as the grouping of these data elements into arrays or structures, if applicable.

12.5.3.1.3 Alternate Access

This parameter shall specify the access which is desired to the data elements of the type specified by the variable specification. If omitted, then full access is desired. If included, then alternate access is desired.

This parameter shall be omitted if Variable Specification references a single simple data element.

12.5.4 Address Parameter

The Address parameter is used to specify an Unnamed Variable object. MMS provides three forms of implementation-defined addresses, as shown by the structure of the Address parameter as given in Table 43.

Table 43 – Address parameter

Conformance: VADR Parameter Name	Req Rsp	Ind Cnf	CBB
Kind Of Address	M	M(=)	
Numeric Address	S	S(=)	
Symbolic Address	S	S(=)	
Unconstrained Address	S	S(=)	

12.5.4.1 Kind Of Address

This parameter shall indicate the kind of address contained in the parameter. Its value shall indicate whether the address is a numeric address, a symbolic address or an unconstrained address.

NOTE – MMS does not specify the relationship between the Kind of Address and the characteristics of a real system. The difference between the various kinds of address, as far as this part of ISO/IEC 9506 is concerned, is purely syntactic. An implementation may support zero or more of these kinds of addresses.

12.5.4.2 Numeric Address

This parameter, of type unsigned integer, shall be selected when the Kind Of Address parameter specifies a numeric address. Its use is appropriate for addressing variables in a system which specifies a linear address space or in which addresses can be represented by non-negative integer values.

12.5.4.3 Symbolic Address

This parameter, of type character string, shall be selected when the Kind Of Address parameter specifies a symbolic address. Its use is appropriate for addressing symbolically named built-in variables.

NOTE – This kind of address, like the other two, specifies the Address attribute of an Unnamed Variable. It is semantically and syntactically distinct from an MMS Variable Name.

12.5.4.4 Unconstrained Address

This parameter, of type octetstring, shall be selected when the Kind Of Address parameter specifies an unconstrained address. Its use is appropriate for addressing variables in a system having an implementation-specific address format which may not be represented as a relative address or as a symbolic address.

12.6 Read Service

The read service is used by a client MMS-user in order to request that a VMD return the value of one or more variables defined at the VMD.

12.6.1 Structure

The structure of the component service primitives is shown in Table 44.

Table 44 – Read Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Specification With Result	M	M(=)			
Variable Access Specification	M	M(=)			
Result(+)			S	S(=)	
Variable Access Specification			C	C(=)	
List Of Access Result			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.6.1.1 Argument

This parameter shall convey the service specific parameters for the Read service request.

12.6.1.1.1 Specification With Result

This boolean parameter shall indicate whether (true) or not (false) the Variable Access Specification parameter is requested in the Result(+) parameter of the response primitive, if issued. If true, and the response primitive specifies success, then the value of the Variable Access Specification parameter of the indication primitive shall be returned in the Result(+) parameter of the response primitive. If false the Variable Access Specification parameter shall not be included.

12.6.1.1.2 Variable Access Specification

This parameter shall specify the variables which are to be accessed. The Variable Access Specification parameter is described in 12.5.1.

12.6.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded.

NOTE – For the Read service, a successful result means that the service request was acceptable to the VMD and that the VMD has attempted to determine the value of each of the variables requested by the service.

When success is indicated the following additional parameters are provided.

12.6.1.2.1 Variable Access Specification

This parameter shall be present if requested in the indication primitive. Otherwise, it shall be omitted. If included, it shall contain the Variable Access Specification parameter from the indication primitive.

12.6.1.2.2 List Of Access Result

This parameter shall contain the values of the specified variables, in the order specified by the Variable Access Specification parameter. Each element of the list shall be an Access Result, which shall either specify the value of the real variable at the time of access, after applying the variable's type description and alternate access (if applicable), or the value of a constructed variable specified by a Scattered Access object, or a reason for access error. The Access Result parameter is described in 12.4.1.

12.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request is invalid or unacceptable to the VMD, and has been denied. The Error Type parameter, which is described in detail in clause 17, provides the reason that the request has been denied.

12.6.2 Service Procedure

After verifying the validity of the service request, the VMD shall attempt to read (see V-Get in 12.1.1.2.1) the values of the specified variables, and shall return, in the order specified in the Variable Access Specification parameter, the Access Result for each read attempt.

12.7 Write Service

The Write service is used by a client MMS-user to request that the VMD replace the content of one or more variables with values supplied in the request.

12.7.1 Structure

The structure of the component service primitives is shown in Table 45.

Table 45 – Write Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Variable Access Specification	M	M(=)			
List Of Data	M	M(=)			
Result(+)			S	S(=)	
List Of Write Result			M	M(=)	
Success			M	M(=)	
Data Access Error			C	C(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.7.1.1 Argument

This parameter shall convey the service specific parameters for the Write service request.

12.7.1.1.1 Variable Access Specification

This parameter shall specify the variable or variables which are to be written. This parameter is described in detail in 12.5.1.

12.7.1.1.2 List Of Data

This parameter shall specify the values to be written to the variables specified by the Variable Access Specification parameter. The values shall occur in this list in the order of the variables specified in the Variable Access Specification parameter. The parameters of Data shall be determined by the variable's type description (or scattered access description) and alternate access description, as applicable. (See 12.2 and 12.3 for details).

The Data parameter is described in 12.4.2.

12.7.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded.

NOTE – For the Write service, a successful result means that the service request was acceptable to the VMD and that the VMD has attempted to replace the value of each of the specified variables with the values supplied in the request.

When success is indicated the List Of Write Result parameter is also included.

12.7.1.2.1 List Of Write Result

The List Of Write Result parameter shall return a list, specified in the order of the variables identified in the request. This list shall indicate, for each variable, either a confirmation that the write to that variable succeeded or a reason that the write to that variable failed.

12.7.1.2.1.1 Success

This boolean parameter shall indicate, for a given variable, whether the write succeeded (true) or not (false).

12.7.1.2.1.2 Data Access Error

If failure is indicated (Success equals false), this parameter shall provide the reason for failure. The description for Data Access Error is found in 12.4.3.

12.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request is invalid or unacceptable to the VMD, and has been denied. The Error Type parameter, which is described in detail in clause 17, provides the reason that the request has been denied.

12.7.2 Service Procedure

The VMD shall verify the validity of the service request by inspecting the entire Variable Access Specification and the List of Data. If the elements of the List of Data do not agree with the Variable Access Specification in type and number of elements, a Result(-) shall be returned. Otherwise, the VMD shall attempt to write (see V-Put in 12.1.1.2.2) the values of each of the specified variables, and then shall return, in the order specified in the Variable Access Specification parameter, a confirmation that the write succeeded or an indication of why the write failed. A Result(+) with a List of Write Result shall be returned.

12.8 InformationReport Service

The InformationReport service may be used by an MMS-user in order to inform the other MMS-user of the value of one or more specified variables, as read by the issuing MMS-user.

12.8.1 Structure

The structure of the component service primitives is shown in Table 46.

Table 46 – InformationReport Service

Parameter Name	Req	Ind	CBB
Argument	M	M(=)	
Variable Access Specification	M	M(=)	
List Of Access Result	M	M(=)	

12.8.1.1 Argument

This parameter shall convey the service specific parameters for the InformationReport service request.

12.8.1.1.1 Variable Access Specification

This parameter shall identify the variables for which values are being reported. This parameter is fully described in 12.5.1.

12.8.1.1.2 List Of Access Result

This parameter shall contain the values of the specified variables, in the order specified by the Variable Access Specification parameter. Each element of the list shall be an Access Result, which shall either specify the value of the variable at the time of access, or a reason for failure. The Access Result parameter is described in 12.4.1.

12.8.2 Service Procedure

MMS does not specify a procedure for invoking, or for receiving, the InformationReport service. The use of this service is application determined. An InformationReport.request shall not be sent if the peer MMS-user did not indicate support of the InformationReport service in the services supported parameter in the Initiate service.

This is an unconfirmed service.

NOTE 1 – The choice of associations (if more than one exists) on which to send the InformationReport service request is a local matter (which may be further specified by Companion Standards). All associations, one association, or some group may be selected.

NOTE 2 – The use of this service is functionally equivalent to an Event Notification with an Event Action of the Read service (with a Specification With Result parameter set to true). The practical difference is that by using the Event Notification method, the conditions under which the service is used are directly visible (and modifiable) using MMS services, while by using the InformationReport service, the conditions are a local matter and cannot be determined or changed by the remote MMS user.

12.9 GetVariableAccessAttributes Service

The GetVariableAccessAttributes service is used by a client MMS-user in order to request that a VMD return the attributes of a Named Variable or an Unnamed Variable object defined at the VMD, or that a VMD return the derived type description of a Scattered Access object defined at the VMD.

12.9.1 Structure

The structure of the component service primitives is shown in Table 47.

Table 47 – GetVariableAccessAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Kind Of Variable	M	M(=)			
Name	S	S(=)			VNAM
Address	S	S(=)			VADR
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Address			C	C(=)	VADR
Type Specification			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.9.1.1 Argument

This parameter shall convey the service specific parameters for the GetVariableAccessAttributes service request.

12.9.1.1.1 Kind Of Variable

This parameter shall equal NAMED or UNNAMED depending upon whether the request is for a named object (Named Variable or Scattered Access), or an Unnamed Variable object, respectively.

12.9.1.1.2 Name

When Kind Of Variable is equal to NAMED, this parameter, of type Object Name, shall specify the Variable Name or Scattered Access Name attribute (as applicable) of the desired object.

12.9.1.1.3 Address

When Kind Of Variable is equal to UNNAMED, this parameter shall specify the Address attribute of the desired Unnamed Variable object.

12.9.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following additional parameters shall also apply.

12.9.1.2.1 MMS Deletable

This boolean parameter shall indicate whether (true) or not (false) the referenced object may be deleted by use of Delete-VariableAccess service. It shall be the MMS Deletable attribute of the referenced object.

12.9.1.2.2 Address

If the VADR parameter conformance building block is in effect for the current application association, and if the referenced object is a Named Variable object with Access Method equal to PUBLIC, this parameter shall be the Address attribute of the Named Variable object. Otherwise, this parameter shall be omitted.

12.9.1.2.3 Type Specification

This parameter shall be the Type Description attribute of the referenced Named Variable or Unnamed Variable object or it shall be the derived type description for the referenced Scattered Access object, as applicable.

12.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.9.2 Service Procedure

The VMD shall locate the specified object and return its attributes or derived type description, as applicable.

12.10 DefineNamedVariable Service

The purpose of the DefineNamedVariable service is to allow a client MMS-user to request that the VMD create a Named Variable object, describing a mapping to a real variable in the VMD.

NOTE – The ability to define a Named Variable object, using MMS services, has been included in this part of ISO/IEC 9506 in order to support those systems which, due to simplicity or age, do not provide for local definition of Named Variable objects.

12.10.1 Structure

The structure of the component service primitives is shown in Table 48.

Table 48 – DefineNamedVariable Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Variable Name	M	M(=)			
Address	M	M(=)			
Type Specification	U	U(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.10.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedVariable service request.

12.10.1.1.1 Variable Name

The Variable Name parameter, of type Object Name, shall specify the name which shall uniquely identify the Named Variable object at the VMD. This name shall be unique among defined Variable Name and Scattered Access Name attributes having the specified scope.

12.10.1.1.2 Address

This parameter shall specify the Address attribute of an Unnamed Variable object. It shall specify the base, or starting, address for the variable described by the Type Specification parameter. If multiple addresses are required to represent this type, they shall be assigned to contiguous locations of the VMD. The Address parameter is described in 12.5.4.

12.10.1.1.3 Type Specification

This parameter is optional. When specified, it defines the Type Description attribute of the Named Variable object which is being defined. If not specified, the Named Variable object inherits the Unnamed Variable object's Type Description attribute. The Type Specification parameter is described in 12.2.

The simple data elements of the type described by the Type Specification shall be compatible with the Type Description attributes of the Unnamed Variable objects which are spanned by this definition.

NOTE – The criteria for deciding that types are compatible are local issues. See description of Address attribute in 12.1.1.5.1.

12.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

12.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.10.2 Service Procedure

If the requested definition is acceptable, the VMD shall create a Named Variable object and shall initialize its attributes as described below.

a) The Variable Name attribute shall be initialized to equal the Variable Name parameter.

b) The MMS Deletable attribute shall be initialized to true.

c) If the Type Specification parameter is present in the indication primitive, the Named Variable object's Type Description attribute shall be initialized to the value of the Type Specification parameter, with all Type Name sub-parameters, if any, replaced by the Type Description attribute of the Named Type object having Type Name attribute equal to the value of the Type Name sub-parameter.

d) If the Type Specification parameter is absent from the indication primitive, the Named Variable object's Type Description attribute shall be initialized to equal the Type Description attribute of the Unnamed Variable object having Address attribute equal to the specified Address parameter.

e) The Access Method attribute shall be initialized to PUBLIC.

f) The Address attribute shall be initialized to the value of the Address parameter.

Finally, a response specifying Result(+) shall be issued. This response shall contain no service-specific information.

12.11 DefineScatteredAccess Service

The purpose of the DefineScatteredAccess service is to allow a client MMS-user to request that the VMD create a Scattered Access object, describing variable access through a structured "variable" constructed from components which are represented by Named Variable, Unnamed Variable, or Scattered Access objects, in any combination.

NOTE – The ability to define a Scattered Access object has been included in this part of ISO/IEC 9506 in order to support those real systems which, due to simplicity or age, do not provide for local definition of Named Variable objects having an Access Method allowing a variable's value to be determined apart from the specification of a single base address, and which do not provide facilities which allow for the server application to arrange the data elements of a variable so that they may be accessed using a single base address.

12.11.1 Structure

The structure of the component service primitives is shown in Table 49.

Table 49 – DefineScatteredAccess Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scattered Access Name	M	M(=)			
Scattered Access Description	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.11.1.1 Argument

This parameter shall convey the service specific parameters for the DefineScatteredAccess service request.

12.11.1.1.1 Scattered Access Name

The Scattered Access Name parameter, of type Object Name, shall specify the name which shall uniquely identify the Scattered Access object at the VMD. This name shall be unique among defined Variable Name and Scattered Access Name attributes having the specified scope.

12.11.1.1.2 Scattered Access Description

The Scattered Access Description parameter (described in 12.5.3) shall specify the component variables (one or more) which are to be accessed using the Scattered Access object.

12.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

12.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.11.2 Service Procedure

If the requested definition is acceptable, the VMD shall create a Scattered Access object. The newly created object's Scattered Access Name attribute shall be initialized to equal the Scattered Access Name parameter, its MMS Deletable attribute shall be initialized to true, and its List Of Component attribute shall be initialized to contain a list of references to the variable access objects specified in the Scattered Access Description parameter.

The entries of the List Of Component attribute shall be ordered according to the order of entries in the List Of Component parameter of the Scattered Access Description parameter. Each entry shall have its attributes initialized as specified below.

a) The Kind Of Reference attribute and the Reference attribute shall be initialized based on the value of the Variable Specification parameter, as follows:

   1) If the Variable Specification parameter specifies a Name, the referenced object (Named Variable or Scattered Access) shall determine the value of Kind Of Reference attribute and the Reference attribute shall be initialized to refer to this object.

   2) If the Variable Specification parameter specifies an Address, the Kind Of Reference attribute shall be initialized to indicate that an Unnamed Variable is referenced and the Reference attribute shall be initialized to refer to this object.

   3) If the Variable Specification parameter specifies a Variable Description, a Named Variable object shall be created and initialized as specified below and then the Kind Of Reference attribute shall be initialized to indicate that a Named Variable object is referenced and the Reference attribute shall be initialized to refer to the newly created object. The attributes of Named Variable object are initialized as follows:

      i) the Variable Name attribute shall be initialized to equal UNDEFINED;

      ii) the MMS Deletable attribute shall be initialized to equal true;

      iii) the Type Description attribute shall be initialized to equal the type specified by the Variable Description parameter's Type Specification;

      iv) the Access Method attribute shall be initialized to equal PUBLIC;

      v) the Address attribute shall be initialized to equal the Variable Description parameter's Address.

   4) If the Variable Specification parameter specifies a Scattered Access Description, the Kind Of Reference attribute shall be initialized to indicate that a Scattered Access object is referenced, then a Scattered Access object shall be created and initialized as specified below, and the Reference attribute shall be initialized to refer to the newly created object. The Scattered Access object's attributes shall be initialized as follows:

      i) the Scattered Access Name attribute shall be initialized to equal UNDEFINED;

      ii) the MMS Deletable attribute shall be initialized to equal true;

      iii) the List Of Component attribute shall be initialized, based on the recursively specified Scattered Access Description parameter, by recursive execution of steps one (1), two (2) and three (3) of this procedure.

b) The Component Name attribute shall be initialized to UNDEFINED if this component of the Scattered Access Description does not contain the Component Name parameter. Otherwise, it shall be initialized to the value of the specified Component Name parameter.

c) The Access Description attribute shall be initialized to UNDEFINED if the Alternate Access parameter is absent. Otherwise it shall be initialized to the value of the Alternate Access parameter.

Finally a response primitive specifying Result(+) shall be issued.

12.12 GetScatteredAccessAttributes Service

The purpose of the GetScatteredAccessAttributes service is to allow a client MMS-user to request that the VMD return the attributes of a Scattered Access object defined at the VMD.

NOTE – The GetVariableAccessAttributes service may be used to obtain the derived Type Specification of a Scattered Access object.

12.12.1 Structure

The structure of the component service primitives is shown in Table 50.

Table 50 – GetScatteredAccessAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scattered Access Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Scattered Access Description			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.12.1.1 Argument

This parameter shall convey the service specific parameters for the GetScatteredAccessAttributes service request.

12.12.1.1.1 Scattered Access Name

The Scattered Access Name parameter, of type Object Name, shall specify the name of the Scattered Access object whose attributes are being requested.

12.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following additional parameters.

12.12.1.2.1 MMS Deletable

This parameter shall be the MMS Deletable attribute of the referenced Scattered Access object.

12.12.1.2.2 Scattered Access Description

The Scattered Access Description parameter (described in 12.5.3) shall specify the component variables (one or more) which are accessed using the Scattered Access object.

12.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

If the Scattered Access Name parameter references a Named Variable object instead of a Scattered Access object, then Error Type shall specify Error Class equal to ACCESS and Error Code equal to OBJECT-NON-EXISTENT.

12.12.2 Service Procedure

The VMD shall locate the specified Scattered Access object and shall return the value of its MMS Deletable attribute as the MMS Deletable parameter and its List Of Component attribute as the List Of Component parameter of the Scattered Access Description parameter as described below.

Entries shall be placed in the List Of Component parameter in the order specified in the List Of Component attribute of the Scattered Access object. Each entry shall correspond to a component of the Scattered Access object's List Of Component attribute and shall have its parameters specified as follows:

a) The Component Name parameter shall be omitted if the Component Name attribute is equal to UNDEFINED. Otherwise, it shall be the Component Name attribute.

b) The parameters of the Variable Specification parameter shall be determined by the values of the Kind Of Reference attribute and by the values of the attributes of the object referenced by the Reference attribute, as follows:

   1) If Kind Of Reference equals NAMED and the referenced Named Variable object's Variable Name attribute does not have the value UNDEFINED, then Kind Of Variable shall be NAMED and Name shall contain the Variable Name attribute of the referenced object.

   2) If the Kind Of Reference equals UNNAMED, then Kind Of Variable shall be UNNAMED, and Address shall contain the Address attribute of the referenced object.

   3) If Kind Of Reference equals SCATTERED and the referenced Scattered Access object's Scattered Access Name attribute does not have the value UNDEFINED, then Kind Of Variable shall be NAMED and Name shall contain the Scattered Access Name attribute of the referenced object.

   4) If Kind Of Reference equals NAMED and the referenced Named Variable object's Variable Name attribute is equal to UNDEFINED, then Kind Of Variable shall be SINGLE and Variable Description shall contain Address equal to the Address attribute of the referenced Named Variable object and Type Specification equal to the referenced Named Variable object's Type Description attribute.

   5) If Kind Of Reference equals SCATTERED and the referenced Scattered Access object's Scattered Access Name attribute is equal to UNDEFINED, then Kind Of Variable shall be SCATTERED, and the Scattered Access Description parameter shall be specified by recursively applying steps one (1) through three (3) of this service procedure for the referenced Scattered Access object.

c) The Alternate Access parameter shall be omitted if the Access Description attribute is equal to UNDEFINED. Otherwise it shall be the Access Description attribute.

Finally a response primitive specifying Result(+) shall be issued.

12.13 DeleteVariableAccess Service

The purpose for the DeleteVariableAccess service is to allow a client MMS-user to request that a VMD delete one or more Named Variable or Scattered Access objects having MMS Deletable attribute equal to true.

12.13.1 Structure

The structure of the component service primitives is shown in Table 51.

Table 51 – DeleteVariableAccess Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scope Of Delete	M	M(=)			
List Of Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

12.13.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteVariableAccess service request.

12.13.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of delete to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific MMS Deletable Named Variable objects or Scattered Access objects (in any combination) having Variable Name or Scattered Access Name attributes (as applicable) equal to the Name parameters of the List Of Name parameter are to be deleted.

AA-SPECIFIC - Specifies that all MMS Deletable Named Variable and Scattered Access objects within the scope of the current application association are to be deleted.

DOMAIN - Specifies that all MMS Deletable Named Variable and Scattered Access objects within the scope of the specified domain are to be deleted.

VMD - Specifies that all MMS Deletable Named Variable and Scattered Access objects having VMD scope are to be deleted.

12.13.1.1.2 List Of Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Name parameters, of type Object Name, each specifying the name attribute (Variable Name or Scattered Access Name) of a specific named variable access object which is to be deleted.

12.13.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It gives the name of the domain for which all MMS Deletable Named Variable and Scattered Access objects are to be deleted.

12.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following additional parameters shall also apply.

12.13.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that matched the name specification in the service request.

12.13.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that were deleted as a result of executing the service procedure.

NOTE – The difference between the Number Matched and Number Deleted parameters indicate the number of objects which were not deleted, either because they have the MMS Deletable attribute equal to false, or for other reasons.

12.13.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

12.13.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that were deleted as a result of executing the service procedure.

12.13.2 Service Procedure

If the request is acceptable, for each Named Variable or Scattered Access object having a Variable Name or Scattered Access Name attribute of the specified scope (or for the specified objects in the case of SPECIFIC) and having MMS Deletable attribute equal to true, the VMD shall delete the object and make its name (Variable Name or Scattered Access Name) available for immediate redefinition.

If a deleted object is referenced by one or more Named Variable List or Scattered Access objects, the Reference attribute of each referencing object shall be set to UNDEFINED.

If a Scattered Access object is deleted, then all referenced Named Variable or Scattered Access objects having name attribute (Variable Name or Scattered Access Name) equal to UNDEFINED shall also be deleted. This procedure shall be repeated for each deleted Scattered Access object. Any such referenced objects deleted shall not be included in the count of the number matched or the number deleted.

After all objects of the specified scope have been deleted, a positive response is issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, then a negative response shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object with the MMS Deletable attribute equal to false shall not be deemed an error.

12.14 DefineNamedVariableList Service

The purpose of the DefineNamedVariableList service is to allow a client MMS-user to request that the VMD create a Named Variable List object, describing variable access through a list of Named Variable objects or Unnamed Variable objects, or Scattered Access objects, in any combination.

12.14.1 Structure

The structure of the component service primitives is shown in Table 52.

Table 52 – DefineNamedVariableList Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Variable List Name	M	M(=)			
List Of Variable	M	M(=)			
Variable Specification	M	M(=)			
Alternate Access	U	U(=)			VALT
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.14.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedVariableList service request.

12.14.1.1.1 Variable List Name

The Variable List Name parameter, of type Object Name, shall specify the name which shall uniquely identify the Named Variable List object at the VMD.

12.14.1.1.2 List Of Variable

The List Of Variable parameter shall specify a list of one or more variables which are to be accessed using the Named Variable List. Each element of this list shall specify the following parameters.

12.14.1.1.2.1 Variable Specification

This parameter shall specify a variable which is to be accessed by this element of the list. The Variable Specification parameter is described in 12.5.2.

12.14.1.1.2.2 Alternate Access

This optional parameter, if included, shall specify Alternate Access which is to apply when the variable specified by this element of the list is accessed. If omitted, full access is specified. Alternate Access is described in 12.3.

12.14.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

12.14.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.14.2 Service Procedure

If the requested definition is acceptable, the VMD shall create a Named Variable List object and shall initialize its attributes as follows.

The Variable List Name attribute shall be initialized to equal the Variable List Name parameter.

The MMS Deletable attribute shall be initialized to true.

The List Of Variable attribute shall be initialized to contain a list of references and access descriptions to the specified variable access objects. The entries shall be placed in the List Of Variable attribute in the order in which they occur in the List Of Variable parameter. Each entry shall correspond to an element of the List Of Variable parameter and shall have its attributes initialized as specified below.

a) The Kind Of Reference attribute and the Reference attribute shall be initialized, based on the value of the Variable Specification parameter, as follows.

   1) If the Variable Specification parameter specifies a Name, the referenced object (Named Variable or Scattered Access) shall determine the value of Kind Of Reference attribute and the Reference attribute shall be initialized to refer to this object.

   2) If the Variable Specification parameter specifies an Address, the Kind Of Reference attribute shall be initialized to indicate that an Unnamed Variable is referenced and the Reference attribute shall be initialized to refer to this object.

   3) If the Variable Specification parameter specifies a Variable Description, a Named Variable object shall be created with its Variable Name attribute equal to UNDEFINED, its MMS Deletable attribute equal to true, its Type Description attribute equal to the type specified by the Variable Description parameter's Type Specification, its Access Method attribute equal to PUBLIC, and its Address attribute equal to the Type Description parameter's Address. Then the Kind Of Reference attribute shall be initialized to indicate that a Named Variable object is referenced and the Reference attribute shall be initialized to refer to the newly created object.

   4) If the Variable Specification parameter specifies a Scattered Access Description, the Kind Of Reference attribute shall be initialized to indicate that a Scattered Access object is referenced, a Scattered Access object shall be created and initialized as specified below, and the Reference attribute shall be initialized to refer to the newly created object. The Scattered Access object's attributes shall be initialized as follows:

      i) the Scattered Access Name attribute shall be initialized to equal UNDEFINED;

      ii) the MMS Deletable attribute shall be initialized to equal true;

      iii) the List Of Components attribute shall be initialized to contain the specified components, each having attributes initialized as follows:

      – Kind Of Reference and Reference shall be initialized as specified by step one (1) of this service procedure, based on the value of the Variable Specification parameter of this component of the Scattered Access Description;

      – Component Name shall be initialized to UNDEFINED if this component of the Scattered access Description does not contain the Component Name parameter. Otherwise, it shall be initialized to the value of the specified Component Name parameter;

      – Access Description shall be initialized to UNDEFINED if this component of the Scattered Access Description does not contain the Alternate Access parameter. Otherwise, it shall be initialized to the value of the specified Alternate Access parameter.

b) The Access Description attribute shall be initialized to UNDEFINED if the Alternate Access parameter is absent. Otherwise it shall be initialized to the value of the Alternate Access parameter.

Finally a response primitive specifying Result(+) shall be issued.

12.15 GetNamedVariableListAttributes Service

The GetNamedVariableListAttributes service is provided in order that a client MMS-user may request that a VMD return the attributes of a Named Variable List object defined at the VMD.

12.15.1 Structure

The structure of the component service primitives is shown in Table 53.

Table 53 – GetNamedVariableListAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Variable List Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
List Of Variable			M	M(=)	
Variable Specification			M	M(=)	
Alternate Access			C	C(=)	VALT
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.15.1.1 Argument

This parameter shall convey the service specific parameters for the GetNamedVariableListAttributes service request.

12.15.1.1.1 Variable List Name

The Variable List Name parameter, of type Object Name, shall specify the Variable List Name attribute of the Named Variable List object whose attributes are desired.

12.15.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following additional parameters shall also apply.

12.15.1.2.1 MMS Deletable

This parameter shall indicate whether (true) or not (false) the Named Variable List object is deletable using the DeleteNamedVariableList service. It shall be the MMS Deletable attribute of the referenced Named Variable List object.

12.15.1.2.2 List Of Variable

The List Of Variable parameter shall be the List Of Variable attribute of the referenced Named Variable List object. It shall specify an ordered list of one or more access descriptions, each specifying the Variable Specification and (conditionally) Alternate Access parameters which specify access to an MMS variable or a Scattered Access object.

12.15.1.2.2.1 Variable Specification

The Variable Specification parameter shall specify the variable (or Scattered Access object) which is accessed by this element of the list. Variable Specification is described in 12.5.2.

12.15.1.2.2.2 Alternate Access

This parameter shall be omitted if full access to the referenced variable is provided. Otherwise, it shall specify the value of the Access Description attribute of this list element. Alternate Access is described in 12.3.

12.15.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.15.2 Service Procedure

The VMD shall locate the specified Named Variable List object and shall return the value of its MMS Deletable attribute as the MMS Deletable parameter and its List Of Variable attribute as the List Of Variable parameter, as specified below.

Entries shall be placed in the List Of Variable parameter in the order specified in the List Of Variable attribute of the Named Variable List object. Each entry shall correspond to an element of the Named Variable List object's List Of Variable attribute and shall have its parameters specified as follows:

a) The parameters of the Variable Specification parameter shall be determined by the values of the Kind Of Reference attribute and then by the values of the attributes of the object referenced by the Reference attribute, as follows:

   1) If Kind Of Reference equals NAMED and the referenced Named Variable object's Variable Name attribute does not have the value UNDEFINED, then Kind Of Variable shall be NAMED and Name shall contain the Variable Name attribute of the referenced object.

   2) If the Kind Of Reference equals UNNAMED, then Kind Of Variable shall be UNNAMED, and Address shall contain the Address attribute of the referenced object.

   3) If Kind Of Reference equals SCATTERED and the referenced Scattered Access object's Scattered Access Name attribute does not have the value UNDEFINED, then Kind Of Variable shall be NAMED and Name shall contain the Scattered Access Name attribute of the referenced object.

   4) If Kind Of Reference equals NAMED and the referenced Named Variable object's Variable Name attribute is equal to UNDEFINED, then Kind Of Variable shall be SINGLE and Variable Description shall contain Address equal to the Address attribute of the referenced Named Variable object and Type Specification equal to the referenced Named Variable object's Type Description attribute.

   5) If Kind Of Reference equals SCATTERED and the referenced Scattered Access object's Scattered Access Name attribute is equal to UNDEFINED, then Kind Of Variable shall be SCATTERED, and the Scattered Access Description parameter shall be specified by recursively applying steps one (1) and two (2) of this service procedure for the referenced Scattered Access object.

b) The Alternate Access parameter shall be omitted if the Access Description attribute is equal to UNDEFINED. Otherwise, it shall be the Access Description attribute.

Finally a response primitive specifying Result(+) shall be issued.

12.16 DeleteNamedVariableList Service

The DeleteNamedVariableList service is provided in order to allow a client MMS-user to request that a VMD delete one or more MMS-defined Named Variable List objects at the VMD having MMS Deletable attribute equal to true.

12.16.1 Structure

The structure of the component service primitives is shown in Table 54.

Table 54 – DeleteNamedVariableList Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List Of Variable List Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

12.16.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteNamedVariableList service request.

12.16.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of delete to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific MMS-defined Named Variable List objects having Variable List Name attributes equal to the Variable List name parameters of the List Of Variable List Name parameter are to be deleted.

AA-SPECIFIC - Specifies that all MMS-defined Named Variable List objects within the scope of the current application association are to be deleted.

DOMAIN - Specifies that all MMS-defined Named Variable List objects within the scope of the specified domain are to be deleted.

VMD - Specifies that all MMS-defined Named Variable List objects having VMD scope are to be deleted.

12.16.1.1.2 List Of Variable List Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Variable List Name parameters, of type Object Name, each specifying the value of the Variable List Name attribute of a specific Named Variable List object which to be deleted.

12.16.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It gives the name of the domain for which all MMS-defined Named Variable List objects are to be deleted.

12.16.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following additional parameters shall also apply.

12.16.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that matched the name specification in the service request.

12.16.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that were deleted as a result of executing the service procedure.

12.16.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

12.16.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable or Scattered Access objects that were deleted as a result of executing the service procedure.

12.16.2 Service Procedure

If the request is acceptable, for each Named Variable List object having a Variable List Name attribute of the specified scope (or for the specified Named Variable List objects in the case of SPECIFIC) and MMS Deletable attribute equal to true, the VMD shall delete the Named Variable List object and make its Variable List Name attribute available for immediate redefinition.

If a deleted Named Variable List object references a Named Variable object having Variable Name attribute equal to UNDEFINED, or a Scattered Access object having a Scattered Access Name attribute equal to UNDEFINED, then the referenced object shall also be deleted. Any such referenced objects deleted shall not be included in the count of the number matched or the number deleted.

If a referenced Scattered Access object is deleted, and it in turn references a Named Variable object having Variable Name attribute equal to UNDEFINED or a Scattered Access object having Scattered Access Name attribute equal to UNDEFINED, then that object shall also be deleted. This procedure shall be repeated for each Scattered Access object, having Scattered Access Name equal to UNDEFINED, which is deleted. Any such referenced objects deleted shall not be included in the count of the number matched or the number deleted.

After all MMS-defined Named Variable List objects of the specified scope have been deleted, a positive response is issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, then a negative response shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object with the MMS Deletable attribute equal to false shall not be deemed an error.

12.17 DefineNamedType Service

The DefineNamedType service is provided so that a client MMS-user may request that a VMD store a named type specification for use in subsequent definition of Named Variable or Named Type (or both) objects at the VMD.

12.17.1 Structure

The structure of the component service primitives is shown in Table 55.

Table 55 – DefineNamedType Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Type Name	M	M(=)			
Type Specification	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.17.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedType service request.

12.17.1.1.1 Type Name

The Type Name parameter, of type Object Name, shall specify the name which shall uniquely identify the Named Type object at the VMD.

12.17.1.1.2 Type Specification

This parameter shall specify the value of the abstract type which shall be associated with the Type Name. A complete description of this parameter can be found in 12.2.1.

12.17.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

12.17.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.17.2 Service Procedure

If the requested definition is acceptable to the VMD, the VMD shall create a Named Type object and initialize its Type Name and Type Description attributes from the Type Name and Type Specification parameters, respectively, and its MMS Deletable attribute to true.

If the Type Specification parameter contains a Type Name parameter referencing a second, defined Named type object, the type description for the type tree node represented by the Type Name parameter shall be inherited from the Type Description attribute of the referenced Named Type object. (A Type Description attribute shall not contain a Type Name). The successful response for the DefineNamedType service shall contain no service-specific information.

12.18 GetNamedTypeAttributes Service

The GetNamedTypeAttributes service is provided so that a client MMS-user may request that a VMD return the attributes of a Named Type object.

12.18.1 Structure

The structure of the component service primitives is shown in Table 56.

Table 56 – GetNamedTypeAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Type Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Type Specification			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

12.18.1.1 Argument

This parameter shall convey the service specific parameters for the GetNamedTypeAttributes service request.

12.18.1.1.1 Type Name

The Type Name parameter, of type Object Name, shall specify the value of the Type Name attribute of the Named Type object whose attributes are desired.

12.18.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the Type Specification parameter shall also be included.

12.18.1.2.1 MMS Deletable

This parameter shall indicate whether (true) or not (false) the Named Type object was created using the DefineNamedType service. It shall be the MMS Deletable attribute of the referenced Named Type object.

12.18.1.2.2 Type Specification

This parameter shall contain the value of the Type Description attribute of the referenced Named Type object. A complete definition of this parameter may be found in 12.2.1.

NOTE – Due to the rules for inheritance of type description for types defined in terms of a Named Type object, the Type Specification parameter of this service will never contain a Type Name parameter.

12.18.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is described in detail in clause 17, provides the reason for failure.

12.18.2 Service Procedure

The VMD shall locate the Named Type object having Type Name attribute equal to the specified Type Name and return the MMS Deletable and Type Description attributes from this object.

12.19 DeleteNamedType Service

The DeleteNamedType service is provided in order to allow a client MMS-user to request that a VMD delete one or more MMS-defined Named Type objects.

12.19.1 Structure

The structure of the component service primitives is shown in Table 57.

Table 57 – DeleteNamedType Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List Of Type Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

12.19.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteNamedType service request.

12.19.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of delete to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific MMS-defined Named Type objects having Type Name attributes equal to the Type Name parameters of the List Of Type Name parameter are to be deleted.

AA-SPECIFIC - Specifies that all MMS-defined Named Type objects within the scope of the current application association are to be deleted.

DOMAIN - Specifies that all MMS-defined Named Type objects within the scope of the specified domain are to be deleted.

VMD - Specifies that all MMS-defined Named Type objects having VMD scope are to be deleted.

12.19.1.1.2 List Of Type Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Type Name parameters, of type Object Name, each specifying the value of the Type Name attribute of a specific Named Type object which to be deleted.

12.19.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It shall be the name of the Domain for which all MMS-defined Named Type objects are to be deleted.

12.19.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following additional parameters shall also apply.

12.19.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Type objects that matched the name specification in the service request.

12.19.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Type objects that were deleted as a result of executing the service procedure.

12.19.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

12.19.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Type objects that were deleted as a result of executing the service procedure.

12.19.2 Service Procedure

If the request is acceptable, for each Named Type object having a Type Name attribute of the specified scope (or for the specified Named Type objects in the case of SPECIFIC) and an MMS Deletable attribute equal to true, the VMD shall delete the Named Type object and make its Type Name attribute available for immediate redefinition.

After all MMS-defined Named Type objects of the specified scope have been deleted, a positive response is issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, then a negative response shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object with the MMS Deletable attribute equal to false shall not be deemed an error.

12.20 Conformance

The Variable Access Services define parameter conformance requirements for the VMD. These requirements are described below.

12.20.1 Parameter Conformance Building Blocks

Parameter conformance to the MMS Variable Access services is specified in terms of the requirements placed upon both the VMD and the client MMS-user. These requirements are defined as follows.

STR1

The STR1 parameter conformance building block shall establish the validity of the ARRAY value for the Kind Of Type parameter of the Type Specification parameter, the INDEX and INDEX-RANGE value for the Access Selection parameters of the Alternate Access parameter (if VALT is supported) and the ARRAY value for the Kind Of Data parameter of the Data parameter.

If STR1 is supported, then these values, and the parameters which they select, are valid, and the NEST parameter conformance building block must specify a value which is greater than zero.

Otherwise, these values, and the parameters which they select, are invalid. A request specifying these values shall constitute a protocol error. A valid request which, if honoured, would require a response specifying these values shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

STR2

The STR2 parameter conformance building block shall establish the validity of the STRUCTURE value for the Kind Of Type parameter of the Type Specification parameter, the COMPONENT value for the Access Selection parameters of the Alternate Access parameter (if VALT is supported) and the STRUCTURE value for the Kind Of Data parameter of the Data parameter.

If STR2 is supported, then these values, and the parameters which they select, are valid, and the NEST parameter conformance building block must specify a value which is greater than zero.

Otherwise, these values, and the parameters which they select, are invalid. A request specifying these values shall constitute a protocol error. A valid request which, if honoured, would require a response specifying these values shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

NEST

The NEST parameter conformance building block shall specify the maximum number of non-leaf nodes of a type tree between the root node and the most deeply nested leaf node. This value shall be zero if neither STR1 nor STR2 are supported. Otherwise it shall be greater than zero and shall apply equally to STR1 and STR2, as applicable.

VNAM

The VNAM parameter conformance building block shall establish the validity of the NAMED value for the Kind Of Variable parameter of the Variable Specification parameter and the NAMED value for the Kind Of Variable parameter of the GetVariableAccessAttributes service's Argument parameter.

If VNAM is supported, then these values, and the parameters which they select, are valid.

Otherwise, these values, and the parameters which they select, are invalid. A request specifying these values shall constitute a protocol error. A valid request which, if honoured, would require a response specifying these values shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

VADR

The VADR parameter conformance building block shall establish the validity of the Address parameter, wherever it occurs in a service table, and the validity of the UNNAMED and SINGLE values for the Kind Of Variable parameter of the Variable Specification parameter, the UNNAMED value for the Kind Of Variable parameter of the GetVariableAccessAttributes service's Argument parameter, and of the value TRUE for the Packed parameter of the Array and Structure parameters of the Type Specification parameter.

If VADR is supported, then the Address parameter and the specified values, and the parameters which they select, are valid.

Otherwise, the Address parameter and the specified values, and the parameters which they select, are invalid. A request specifying these parameters, values, or both, shall constitute a protocol error. A valid request which, if honoured, would require a response specifying these parameters or values, or both, shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

VALT

The VALT parameter conformance building block shall establish the validity of the Alternate Access and Component Name parameters, wherever they occur in a service table.

If VALT is supported, then these parameters are valid.

Otherwise, these parameters are invalid. A request specifying these parameters shall constitute a protocol error. A valid request which, if honoured, would require a response specifying these parameters shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

VSCA

The VSCA parameter conformance building block shall establish the validity of the Scattered Access Description parameter, wherever it occurs in a service table.

If VSCA is supported, then this parameter is valid.

Otherwise, this parameter is invalid. A request specifying this parameter shall constitute a protocol error. A valid request which, if honoured, would require a response specifying this parameter shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

VLIS

The VLIS parameter conformance building block shall establish the validity of the Variable List Name value for the Kind Of Access parameter of the Variable Access Specification parameter.

If VLIS is supported, then this value, and the parameters which it selects, are valid.

Otherwise, this value, and parameters which it selects, are invalid. A request specifying this value shall constitute a protocol error. A valid request which, if honoured, would require a response specifying this value shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

REAL

The REAL parameter conformance building block shall establish the validity of the use of the representation for real values defined in ISO 8824.

If REAL is supported, then variables and types may be defined with a Type Specification which includes the real data type, and the productions defined in ISO/IEC 9506-2 involving this type shall be supported.

Otherwise, this choice of the Type Specification and of the Data production are invalid. A valid request which, if honoured, would result in a response specifying this choice shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED. A definition request specifying this choice shall result in a service error specifying Error Class equal to DEFINITION and Error Code equal to TYPE-UNSUPPORTED.

12.20.2 Static Conformance

The static conformance statement for an implementation shall state the level of support which it provides, if any, for uninterruptibility of access to variables described by Named Variable, Unnamed Variable and Scattered Access objects. Uninterruptibility of access is defined to mean that:

a) read access is performed with the guarantee that the variable is not changing while being read;

b) write access is performed with the guarantee that all potential concurrent access to the variable (whether local or remote, for read or for write) is inhibited during the write.

If an implementation supports uninterruptibility of access for a subset of its variables, then the static conformance statement shall specify the requirements associated with this subset, and shall specify whether or not variables failing to meet these requirements are accessible using MMS services.

In addition to the required statement concerning uninterruptibility of access, an implementation supporting VADR must specify its address format or formats, and must specify the relationship between an address and an inherent type associated with that address.

12.21 Guidance To Implementors

Subclause 12.21 and its subclauses are informative.

The MMS Variable Access services are intended to serve the needs of a wide range of systems, from the most simple to the highly complex. These services are also intended to serve the needs of devices designed with this part of ISO/IEC 9506 in mind, as well as the needs of devices which, due to age or simplicity, were not designed with this part of ISO/IEC 9506 in mind.

Due to this wide range of requirements, not all of the Variable Access services are appropriate for every VMD. It is expected that there will be three primary environments for implementation of MMS Variable access services. They have been classified as follows:

1) VNAM-only - serving the needs of systems designed with MMS in mind;

2) VADR-only - serving the needs of simple systems and of systems which, due to age or other consideration, do not justify VNAM support;

3) VNAM-with-VADR - serving the needs of systems which were not designed with MMS in mind, and for which it is justified, for whatever reason, to bring as much as possible of the VNAM-only functionality to the device.

NOTE – Other environments are possible and are not prohibited.

The conformance issues associated with these three environments are described below.

12.21.1 VNAM-only

A VNAM-only system is one which has been designed with MMS in mind. It directly supports the Variable Access model through the local definition of MMS Named Variable objects by the serving application process in which the VMD exists. These objects have Access Method attribute not equal to PUBLIC. Unnamed Variable and Scattered Access objects are not supported, though the access method associated with a Named Variable object may, in fact, make use of address information or scattered access information, or both.

From a parameter conformance stand-point, this system will support VNAM. It is likely to support both STR1 and STR2 (and thus NEST greater than zero) and it may also support VALT.

12.21.2 VADR-only

The VADR-only system is one which is either quite simple, thus not justifying VNAM-only, or it has not been designed with MMS in mind and does not justify the complexity of the VNAM-with-VADR environment. This system supports Unnamed Variable objects only.

From a parameter conformance stand-point, this system will support VADR. It is unlikely to support either STR1 or STR2 (and thus NEST will be equal to zero and VALT will not be supported). Neither VNAM nor VSCA are supported.

NOTE – Note that while no variable names are supported in this system, names for other objects, such as domains and program invocation, may still be supported.

12.21.3 VNAM-with-VADR

This environment encompasses all MMS-capable systems which implement the Variable Access services and which are not included in the previous environments. The goal of this environment should be to present a VNAM-only view to most clients.

13 Semaphore Management Services

The Semaphore Management clause contains services which allow synchronization, control and coordination of shared resources between MMS users. The mechanism provided by the VMD is to control and relinquish a Semaphore object.

A semaphore is referenced by a VMD-specific or Domain-specific name. It may be predefined, or defined due to the creation of a domain, or by the DefineSemaphore service.

A semaphore may be used by MMS users to control the access to a specific subset of the VMD, or to synchronize remote applications. However, the VMD does not enforce the protection of a specific subset by a semaphore and the rules of utilization of a semaphore in a given application have to be agreed between the user applications.

MMS provides the capability of managing two types of semaphores: Token Semaphores, allowing single or multiple owners, and Pool Semaphores allowing a dynamic or explicit allocation of Named Tokens. A Named Token is an entity referenced by a name, whose semantic is locally defined and cannot be explicitly managed by MMS services.

The application may prevent deadlock by using time outs, either in the responding MMS-user by providing a parameter in the service primitive used for obtaining the control of a semaphore, or in the requesting MMS-user by canceling pending requests.

Services supporting management of semaphores are:

a) TakeControl;

b) RelinquishControl;

c) DefineSemaphore;

d) DeleteSemaphore;

e) ReportSemaphoreStatus;

f) ReportPoolSemaphoreStatus;

g) ReportSemaphoreEntryStatus.

The AttachToSemaphore Modifier may also be used by an MMS-user to request that a responding MMS-user executes the service procedure associated with a specific confirmed MMS-request under the control of a semaphore.

13.1 The Semaphore Management Model

The Semaphore Management Services apply to semaphore objects managed by the VMD. MMS is able to manage two classes of semaphore objects:

a) Token Semaphores;

b) Pool Semaphores.

Each class of semaphore is defined by a state machine and specific attributes.

A semaphore is defined as a specific instantiation of the attributes of a generic semaphore class, and is referenced by a VMD-specific or Domain-specific name. This name follows the same rules as every object name in the VMD. The instantiation of a semaphore may be predefined in the VMD, or result of a domain creation or a DefineSemaphore service.

NOTE – MMS only provides a service for the definition of Token Semaphores. A Pool Semaphore requires a linking between some physical or logical local entities in the real device and the Named Token handled by the Pool Semaphore in the VMD. Typically, a Pool Semaphore is the representation in the VMD of a real pool semaphore controlling real resources either from local or remote access; MMS cannot create objects in the real devices (only in the VMD), and cannot create an explicit mapping between real and virtual resources. For this reason, Pool Semaphores can only be predefined. The creation of a Token Semaphore is provided by MMS with the intent that this semaphore will be used to synchronize applications between MMS-users, but not to ensure an exclusive access to virtual representations of real resources, since a virtual object cannot control the local application. The local environment of the VMD may or may not be aware of the existence of an MMS-defined semaphore.

A semaphore is modelled as a queue processor, a list of owners and a queue of requesters. Each element of the queue is an object called a Semaphore Entry: it is created by a TakeControl request, or any request modified by the AttachToSemaphore Modifier, or by local means, and records the parameters provided by the request. As soon as an element of the queue is served, it is moved into the list of the owners. The two classes of semaphores follow this general model, but each class defines a specific state machine and a specific queue-serving algorithm.

The owner of a semaphore is an Application Process in the OSI environment, including the Application Process local to the VMD. It is identified by an Application Reference. Unless the owner is the local Application Process, it shall maintain the Application Association with the server used for requesting the semaphore until it relinquishes its control. If the Application Association disappears while a semaphore is still under control, the control is either relinquished or enters the HUNG state, depending upon the parameters specified by the requesting MMS-user.

An Application Process may issue multiple control requests on the same Semaphore under the same or multiple Associations and gain multiple ownership of a Token or Pool Semaphore. The server is able to differentiate these ownerships as long as they are not under the same association; however, a third party MMS-user using the MMS services cannot differentiate them since the Application Association is known only by the peer entities.

NOTE – Multiple ownerships of a Pool Semaphore under the same association are differentiable by the Named Token; multiple ownerships of a Token Semaphore under the same association are not differentiable.

13.1.1 The Semaphore Object

Object: Semaphore

```
Key attribute: Semaphore Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Class (TOKEN, POOL)
Constraint: Class = TOKEN
    Attribute: Number Of Tokens
    Attribute: Number Of Owned Tokens
Constraint: Class = POOL
    Attribute: List Of Named Tokens
    Attribute: List Of Named Token States
Attribute: List Of Owners
Attribute: List Of Requesters
Attribute: Event Condition Reference
```

Semaphore Name

The Semaphore Name attribute identifies the semaphore. It shall be either a VMD-specific, or a Domain-specific Object Name.

MMS Deletable

The MMS Deletable attribute is a boolean value that indicates whether or not this Semaphore object may be deleted using the DeleteSemaphore service.

Class

The Class attribute may have the value TOKEN or POOL and specifies the class of the semaphore.

Number Of Tokens

The Number Of Tokens attribute defines the maximum number of owners allowed by a Token Semaphore.

Number Of Owned Tokens

The Number Of Owned Tokens attribute contains the number of tokens currently owned for a Token Semaphore.

List Of Named Tokens

The List Of Named Tokens attribute defines the names of the tokens controlled by a Pool Semaphore.

List Of Named Token States

The List Of Named Token States attribute contains the state FREE or OWNED for each Named Token controlled by a Pool Semaphore.

List Of Owner

The List Of Owners attribute contains a list of references to the Semaphore Entry objects owning the semaphore.

List Of Requester

The List Of Requesters attribute contains a list of references to the Semaphore Entry objects waiting to obtain the control of the Semaphore.

Event Condition Reference

The Event Condition Reference attribute is a reference to an Event Condition object (See 15.1.1) whose Event Condition Name attribute value is equal to the Semaphore object's Semaphore Name attribute value, whose MMS Deletable attribute value is false, whose Event Condition attribute value is equal to NETWORK-TRIGGERED, whose State attribute is DISABLED, whose Priority attribute value is normalPriority, and whose Severity attribute value is normalSeverity,

13.1.2 The Semaphore Entry Object

Object: Semaphore Entry

```
Key attribute: Entry ID
Attribute: Entry Class (SIMPLE, MODIFIER)
Attribute: Semaphore Reference
Attribute: Requester Application Reference
Attribute: Application Association Local Tag
Attribute: Invoke ID
Attribute: Named Token
Attribute: Priority
Attribute: Remaining Acquisition Delay
Attribute: Remaining Control Time Out
Attribute: Abort On Time Out (TRUE, FALSE)
Attribute: Relinquish If Connection Lost (TRUE, FALSE)
Attribute: Entry State (QUEUED, OWNER, HUNG)
```

Entry ID

The Entry ID attribute identifies locally a Semaphore Entry object and shall be unique for all the Semaphore Entry objects related to a Semaphore. This attribute provides a tag to be used in the segmented ReportSemaphoreEntryStatus service.

Entry Class

The Entry Class attribute contains the value MODIFIER if the Semaphore Entry has been created by a service modified by the AttachToSemaphore Modifier, otherwise it contains the value SIMPLE.

Semaphore Reference

The Semaphore Reference attribute identifies by reference the semaphore requested or owned by the Semaphore Entry.

Requester Application Reference

The Requester Application Reference attribute identifies the Application Process whose request created the Semaphore Entry.

Application Association Local Tag

The Application Association Local Tag attribute identifies locally the Application Association under which the Semaphore Entry was created. This attribute cannot be reported through any of the MMS services.

Invoke ID

The Invoke ID attribute identifies a Transaction object in the VMD (see 7.3). This Transaction object exists at least as long as the Semaphore Entry is waiting for the control of the semaphore if the Entry Class is SIMPLE, or as long as the semaphore has not been relinquished if the Entry Class is MODIFIER. The possible nesting of modifiers is managed by this Transaction object.

Named Token

The Named Token attribute is used only if the requested semaphore is a Pool Semaphore. It contains either a requested Named Token, a controlled Named Token, or the value UNDEFINED.

Priority

The Priority attribute specifies the priority that the Semaphore Entry should have compared to other Semaphore Entry objects while in the waiting list. The priority attribute shall be an integer with values ranging from 0 to 127 inclusive. 0 shall represent the highest priority, 64 the normal priority and 127 the lowest priority. Treatment of priority by the VMD is a local matter.

Remaining Acquisition Delay

The Remaining Acquisition Delay attribute shall specify the duration of time that a Semaphore Entry can remain in the Entry State QUEUED and only has meaning if the Entry State attribute has the value QUEUED. The value shall be either a positive number or the value FOREVER.

When a Semaphore Entry Object is created and the initial value of the Remaining Acquisition Delay attribute is not FOREVER, then a Remaining Acquisition Delay Timer associated with this object shall be initialized and activated. This timer, when activated, shall start decrementing from its initial value to zero. The granularity of the decrementation supported by the VMD shall be specified in the static conformance statement of the PICS. The granularity of one millisecond is not required. Upon reaching zero, the acquisition delay shall have expired and the Semaphore Entry Object shall be deleted and its reference from the associated Semaphore's List Of Requesters shall be deleted. Upon transition from the Entry State QUEUED to the Entry State OWNER or deletion of the Semaphore Entry Object, the Remaining Acquistion Delay Timer shall be deactivated and the decrementation process shall cease.

Remaining Control Time Out

The Remaining Control Time Out attribute shall specify the duration of time that a Semaphore Entry can remain in the Entry State OWNER or HUNG. The attribute shall only have meaning if the Entry State attribute has the value OWNER or HUNG. The value is either a positive number or the value FOREVER.

When a Semaphore Entry Object transitions from the Entry State QUEUED to OWNER and the initial value of the Remaining Control Time Out attribute is not FOREVER, then a Remaining Control Time Out Timer associated with this object shall be initialized and activated. The timer, when activated, shall start decrementing from its initial value to zero. The granularity supported supported by the VMD shall be specified in the static conformance statement of the PICS. A granularity of one millisecond is not required. Upon reaching zero, a Control Time Out shall have occurred and the Semaphore Entry Object shall be deleted, its reference from the associated Semaphore's List Of Owners shall be deleted and the Event Condition which is referenced by the Event Condition Reference attribute of the associated Semaphore shall be triggered. Upon deletion of the Semaphore Entry Object the Remaining Control Time Out Timer shall be deactivated and the decrementation process shall cease.

The action of the Remaining Control Time Out Timer shall not be affected if the Semaphore Entry Object transitions into the Entry State HUNG.

Abort On Time Out

The Abort On Time Out attribute is a boolean which specifies whether (true) or not (false) the Application Association shall be aborted if the Control Time Out occurs. The requesting MMS-user may make use of the Event Condition association with the Semaphore together with appropriate Event Actions and Event Enrollments (see clause 15) to define remedial procedures which should be performed following a Control Time Out.

NOTE – There is only one Event Condition for any given Semaphore. This Event Condition reflects the state of all the subordinate Semaphore Entry objects in that if a Control Time Out occurs for any Semaphore Entry object, the Event Condition is triggered. An MMS-user may make use of the ReportSemaphoreEntryStatus service to determine which Semaphore Entry caused the transition; this Entry will be in the HUNG state.

Relinquish If Connection Lost

The Relinquish If Connection Lost attribute specifies if true that the Semaphore Entry shall relinquish the semaphore if the Application Association identified by the Application Association Local Tag attribute is lost. The value false means that, in such a situation, the Semaphore Entry shall not relinquish the semaphore, but rather shall place the value HUNG in the Entry State attribute.

Entry State

The Entry State attribute contains the value QUEUED while the Semaphore Entry is in the waiting list, OWNER while it is in the Owner list and the Application Association is still maintained, and HUNG while it is in the Owner list and the Application Association is lost.

13.1.3 Model of a Semaphore Entry

The Semaphore Entry model is provided in Figure 9.

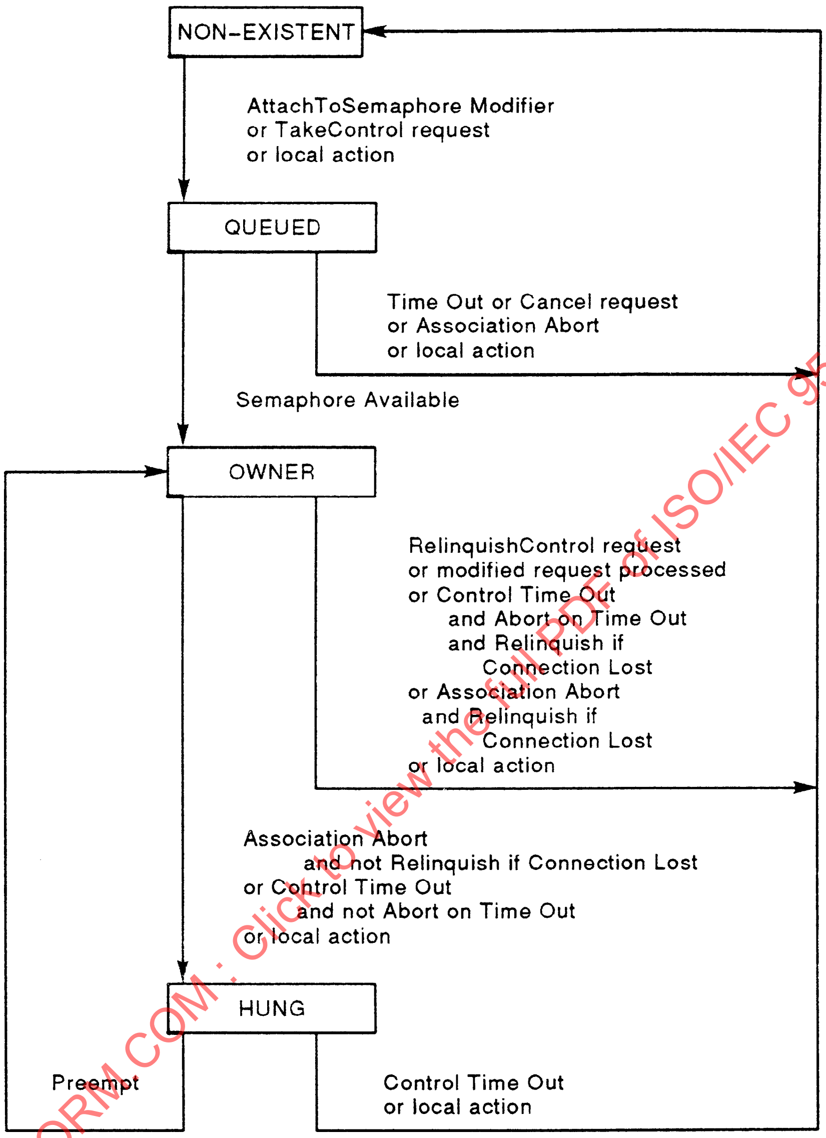

NON-EXISTENT
AttachToSemaphore Modifier or TakeControl request or local action
QUEUED
Time Out or Cancel request or Association Abort or local action
Semaphore Available
OWNER
RelinquishControl request or modified request processed or Control Time Out and Abort on Time Out and Relinquish if Connection Lost or Association Abort and Relinquish if Connection Lost or local action
Association Abort and not Relinquish if Connection Lost or Control Time Out and not Abort on Time Out or local action
HUNG
Preempt
Control Time Out or local action

Figure 9 – Semaphore Entry model

A Semaphore Entry shall be created either by a TakeControl request, or by any MMS request modified by an AttachToSemaphore Modifier, or by a local entity requesting control of the semaphore. After creation, the Semaphore entry shall be placed in a queue.

The queue shall be ordered by an algorithm dependent on the class of semaphore serving the queue. When the Entry is at the top of the queue and the semaphore is able to serve it, the Entry shall be granted control of the semaphore and shall be removed from the queue. If the Entry has been created by a TakeControl request, a positive response shall be issued. If it has been created by a modified request, the modified request shall be released for further processing as specified in the service procedure under control of the Transaction Object.

If the Remaining Acquisition Delay timer associated with the request expires before control has been granted, the Entry shall be deleted and a negative response shall be issued.

An Entry which is in control of the semaphore shall release control of the semaphore either following a RelinquishControl request issued on the same association, or when the modified request has been processed, or by local means, depending upon the way the control had been requested. If a Control Time Out occurs, depending upon the value of the Abort on Time Out attribute, either the association shall be aborted and the Entry processed as for an association abort, or the Entry shall be placed in the HUNG state. The related Event Condition shall be triggered.

If the application association upon which the request was received is aborted while the Entry is in control of the semaphore and the value of the Relinquish If Connection Lost attribute is true, the control of the Semaphore shall be released. If the application association upon which the request was received is aborted while the Entry is in control of the semaphore and the value of the Relinquish If Connection Lost attribute is false, the Entry shall stay in the HUNG state until a preemptive TakeControl request is issued by any MMS-user.

Multiple requests for control of a single semaphore, issued by the same user to the same responding MMS-user are independent. Therefore the second request shall remain queued while the first request is served.

13.1.4 Model of the Token Semaphore

The Token Semaphore is described by the model in Figure 10

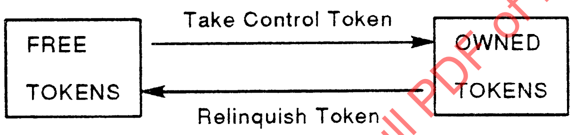


Figure 10 – Token Semaphore model

A Token Semaphore is modeled by a collection of identical tokens, each token evolving between the states FREE and OWNED. FREE TOKENS is the collection of tokens in the state FREE, and OWNED TOKENS the collection of tokens in the state OWNED. The total number of tokens represents the maximum number of owners of the semaphore (a Token Semaphore with only one token is a mutually exclusive semaphore). The state of the semaphore is defined by the number of FREE tokens and the number of OWNED Tokens. The Take Control Token transition moves one token from the state FREE to the state OWNED, while the Relinquish Token transition moves one token from the state OWNED to the state FREE.

Each OWNED token is associated with a Semaphore Entry in the state OWNER or HUNG. The Take Control Token transition models the association of a Token with a Semaphore Entry and the transition of a Semaphore Entry from the state QUEUED to OWNER. The Relinquish Control transition models the deletion of a Semaphore Entry in the state OWNER.

At the creation of the semaphore, all the tokens shall be FREE. As soon as a Semaphore Entry is created, one token shall evolve into the state OWNED. A token shall evolve from FREE to OWNED each time there is a FREE token and there is a Semaphore Entry in the state QUEUED, either following the release of a token or the creation of a Semaphore Entry. A token shall evolve from OWNED to FREE when the associated Semaphore Entry is deleted, following a RelinquishControl request or the end of processing of a AttachToSemaphore modified request, or through a local action. A preempt request shall maintain the token in the state OWNED while changing the associated Semaphore Entry.

The queue of waiting Semaphore Entry shall be served on a first-in-first-out basis for the entries of same priority. The algorithm used for handling prioritized queues is a local issue, and shall be specified in the static conformance statement of the PICS.

13.1.5 Model of the Pool Semaphore

A Pool Semaphore may be described by the following model.

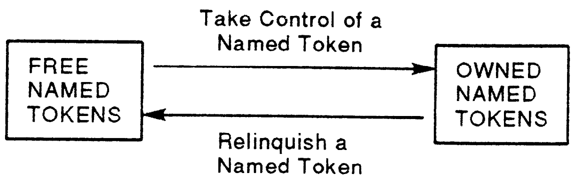


Figure 11 – Pool Semaphore model

A Pool Semaphore is modeled by a collection of Named Tokens, each Named Token evolving between the states FREE and OWNED. The difference between a Token Semaphore and a Pool Semaphore from a modelling point of view is that the tokens handled by a Pool Semaphore are identifiable by a name, which may be specified when requesting the semaphore. With that difference, the description of the Token Semaphore applies to the Pool Semaphore.

A Named Token is the representation in the VMD of some physical or logical entity in the local application. The semantic associated with this entity is known only by the user application and is irrelevant to the VMD; as a consequence, a Named Token cannot identify any object of the VMD, such as a Named Variable or a Domain, and the VMD cannot enforce any semaphore controlled access to an object of the VMD.

NOTE – It is still possible that a Named Token identifies a physical variable in the local device, and that this physical variable may be mapped to an MMS variable. In such an example, the MMS Pool Semaphore may provide a controlled access to the physical variable (assuming the local device will enforce this control), but does not interact with the MMS Variable Access service: The access control will be enforced by the local device (by the V-Get and V-Put functions described in clause 12) , and is not described in the VMD model.

The queue of waiting Semaphore Entries shall be served on a first-in-first-out basis for the entries of same priority. If the entry at the top of the list requests a non-available Named Token, this entry shall remain at the top of the list and the next entry is processed. The algorithm used for handling prioritized queues is a local issue, and shall be specified in the static conformance statement of the PICS.

13.1.6 Operations

The operations on these objects are the following:

TakeControl creates a Semaphore Entry object, adds it to the List Of Requesters of the semaphore and waits until the Semaphore Entry becomes an owner of the semaphore.

TakeControl with preemption replaces the attributes of a specific HUNG Semaphore Entry in the List Of Owners of the semaphore by the value of the parameters provided in the request, and sets its state to OWNER.

RelinquishControl deletes a specified Semaphore Entry in the List Of Owner of a semaphore and modifies the state of the semaphore.

DefineSemaphore creates a Semaphore object.

DeleteSemaphore deletes a Semaphore object.

ReportSemaphoreStatus reports to the requesting MMS-user the attributes of a semaphore.

ReportPoolSemaphoreStatus reports to the requesting MMS-user the attributes of the List Of Named Tokens of a Pool Semaphore. This is a segmented service.

ReportSemaphoreEntryStatus reports to the requesting MMS-user the attributes of the list of Semaphore Entry related to a semaphore, sorted by their state, QUEUED, OWNER or HUNG. This is a segmented service.

AttachToSemaphore Modifier creates a Semaphore Entry object referencing the modified request, and adds it to the List Of Requesters of the specified semaphore. When the Semaphore Entry becomes an owner of the semaphore, it processes the modified request, then deletes the Semaphore Entry. The modified request may include other modifiers, including the AttachToSemaphore Modifier.

13.2 TakeControl Service

The TakeControl service may be used by an MMS-user in order to obtain control of a semaphore.

13.2.1 Structure

The structure of the component service primitives is shown in Table 58.

Table 58 – TakeControl Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Named Token	U	U(=)			
Priority	M	M(=)			
Acceptable Delay	U	U(=)			
Control Time Out	U	U(=)			
Abort On Time Out	C	C(=)			
Relinquish If Connection Lost	M	M(=)			
Application To Preempt	U	U(=)			
Result(+)			S	S(=)	
Named Token			C	C(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

13.2.1.1 Argument

This parameter shall convey the parameters of the TakeControl service request.

13.2.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the semaphore to be controlled.

13.2.1.1.2 Named Token

This optional parameter, of type Identifier, shall be provided only if the semaphore is a Pool Semaphore, and shall not be provided otherwise. If provided, it shall specify the Named Token controlled by the Pool Semaphore over which the MMS-user wishes to be granted control. If this parameter is not provided and if the semaphore is a Pool Semaphore, the choice of the allocated Named Token shall be made by the responding MMS-user.

13.2.1.1.3 Priority

This parameter, of type integer, shall identify the priority that this TakeControl service request should have as compared to (possible) other TakeControl requests, as well as to services using the AttachToSemaphore Modifier, that are awaiting the semaphore identified in the TakeControl request.

13.2.1.1.4 Acceptable Delay

This optional parameter, of type integer, shall indicate the duration of time in milliseconds for which the requesting user is willing to wait for control to be allocated. If a zero value is specified for acceptable delay, this means that no delay is acceptable. (That is, if the semaphore or Named Token is not immediately available, the service shall fail.) If no value is specified for acceptable delay, this shall be interpreted as meaning that any delay is acceptable (i.e. "wait forever"). The granularity of one millisecond is not required. The granularity supported by the VMD shall be specified in the static conformance statement of the PICS.

13.2.1.1.5 Control Time Out

This optional parameter, of type integer, shall specify the duration of time in milliseconds for which control of the semaphore may be held (after it is obtained). If this time limit is exceeded, and the value of the Abort On Time Out parameter is true, the Application Association shall be aborted with a provider abort and the Semaphore Entry shall be relinquished or changed to the HUNG state according to the Relinquish If Connection Lost parameter. If this time limit is exceeded and the value of the Abort On Time Out parameter is false, the Semaphore Entry shall be changed to the HUNG state. In either case, the related Event Condition shall be triggered. If no value is provided, no Control Time Out applies, and the semaphore may be held indefinitely. The granularity of one millisecond is not required. The granularity supported by the VMD shall be specified in the static conformance statement of the PICS.

13.2.1.1.6 Abort On Time Out

This parameter, of type boolean, shall be provided if the Control Time Out parameter is provided. The value true shall specify that the Association shall be aborted in case the Control Time Out expires. The value false shall specify that the related Event Condition shall be signalled.

13.2.1.1.7 Relinquish If Connection Lost

This parameter, of type boolean, shall specify if true that the responding MMS-user shall relinquish the semaphore if the owner of the semaphore loses the ability to control it, either due to the abort of the association used for acquiring the semaphore, or to a local failure. The value false shall mean that in the same situation the responding MMS-user shall maintain the semaphore in the OWNED state, with the associated Semaphore Entry in the HUNG state.

13.2.1.1.8 Application To Preempt

This optional parameter, of type Application Reference, shall specify the owner of a Semaphore Entry which is in the HUNG state. The presence of this parameter shall indicate that the requester of the service wants to take control of a semaphore by preempting control of a Semaphore Entry having state HUNG whose owner matches the Application To Preempt parameter.

13.2.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. A successful result shall return a Named Token parameter if the specified semaphore is a Pool Semaphore, and no parameter if the semaphore is a Token Semaphore. The Named Token is either the optional parameter specified in the request, or a Named Token allocated from the pool of Named Tokens if the parameter was not provided in the request.

13.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.2.2 Service Procedure

If the Application To Preempt parameter is not provided in the indication primitive, the responding MMS-user shall create a Semaphore Entry object and add it to the queue of the semaphore. The attributes of the created Semaphore Entry shall be initialized as follows:

a) The Entry ID attribute shall contain a locally defined value, which shall be unique for the semaphore.

b) The Entry Class attribute shall be initialized to SIMPLE.

c) The Semaphore Name attribute shall be initialized to the value of the Semaphore Name parameter.

d) The Requester Application Reference shall be initialized to the value identifying the Application Process from which the request was issued.

e) The Application Association Local Tag attribute shall be initialized to a value identifying the application association over which the request has been received.

f) The Invoke ID attribute shall be initialized to the value of the Invoke ID parameter of the indication.

g) The Named Token attribute shall be initialized to the value of the Named Token parameter of the indication if provided, otherwise to the value UNDEFINED.

h) The Priority attribute shall be initialized to the value of the Priority parameter.

i) If no value of the Acceptable Delay parameter is provided, then the value of the Remaining Acquisition Delay attribute shall be set to the value FOREVER. If a value of the parameter is provided, then the Acquisition Delay attribute shall be initialized to the value of the Acceptable Delay parameter, and the associated Acquisition Delay Timer activated.

j) If no Control Time Out parameter is provided, then the value of the Remaining Control Time Out attribute shall be set to the value FOREVER. If a value of this parameter is provided, then the Control Time Out attribute shall be initialized to the value of the Control Time Out parameter and the associated Control Timer activated.

k) The Abort On Time Out attribute shall be initialized to the value of the Abort On Time Out parameter.

l) The Relinquish If Connection Lost attribute shall be initialized to the value of the Relinquish If Connection Lost parameter.

m) The Entry State shall be initialized to the value QUEUED.

When the semaphore becomes available, the responding MMS-user shall place the semaphore in a controlled state and return the response service primitive. If the service request is made to a Pool Semaphore, a member of the Named Token pool shall be placed in a controlled state and its name returned in the response. The detailed description of the operations on each class of semaphore may be found in 13.1.

If the Application To Preempt parameter is present in the indication primitive, the responding MMS-user shall check that there is at least one Semaphore Entry matching the Application To Preempt and Named Token parameters in the HUNG state. If there is more than one matching Semaphore Entry, the responding MMS-user shall choose one of them to preempt. If there is no such matching Semaphore Entry, the service fails and an error shall be returned.

If there is one or more Semaphore Entries to preempt, the responding MMS-user shall perform the following:

a) Replace the current values of the Requester Application Reference, Application Association Local Tag, and Relinquish If Connection Lost attributes by the values provided in the parameters of the indication.

b) If the Remaining Control Time Out parameter is not present in the indication, then set the value of the Remaining Control Time Out attribute to the value of FOREVER and deactivate any associated Remaining Control Time Out Timer.

c) If a Remaining Control Time Out parameter is present in the indication, then intialize the Remaining Control Time Out timer to the value in the indication and activate the associated timer.

d) Initialize the Entry State attribute to the value OWNER and the Entry Class attribute to SIMPLE.

e) Return the positive response service primitive.

The service shall fail and return an error for any of the following conditions:

a) the Semaphore Name or the Named Token specified in the service request is unknown or cannot be accessed;

b) the acquisition delay timer has expired or the service request has been cancelled by a Cancel service or a control time out has occurred;

c) a preemptive control has been requested upon a semaphore and there is no Semaphore Entry matching the provided parameters;

d) the VMD cannot process the service request due to a lack of local resources, for example the semaphore queues are full;

e) the service has been cancelled by the VMD in order to prevent a deadlock.

13.3 RelinquishControl Service

The RelinquishControl service may be used by an MMS-user to relinquish control of a semaphore for which control, granted via a previous TakeControl service, is held.

13.3.1 Structure

The structure of the component service primitives is shown in Table 59.

Table 59 – RelinquishControl Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Named Token	C	C(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.3.1.1 Argument

This parameter shall convey the parameters of the RelinquishControl service request.

13.3.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name of the semaphore for which control is to be relinquished.

13.3.1.1.2 Named Token

This parameter, of type Identifier, shall be provided if the semaphore is a Pool Semaphore and shall not be provided otherwise. It shall specify the Named Token to be relinquished and shall be the same as the Named Token from the Result(+) parameter of the TakeControl.Confirm service primitive in which control was granted.

13.3.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. A successful result shall return no service specific parameters.

13.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.3.2 Service Procedure

The responding MMS-user shall release the semaphore from its controlled state by the requesting MMS-user, delete the related Semaphore Entry and return the response service primitive. If the service request was made to a member of a controlled pool of Named Token, the pool member shall be released from the control of the requesting MMS-user.

If the requesting MMS-user has issued multiple successful TakeControl requests to the same semaphore, it shall issue an equal number of RelinquishControl requests to release all held elements of the semaphore.

The service shall fail and return an error for any of the following conditions:

a) the specified semaphore is unknown or cannot be accessed;

b) the specified semaphore is not owned by the requesting MMS-user under the same application association.

13.4 DefineSemaphore Service

The DefineSemaphore service may be used by an MMS-user to create a Token Semaphore at a responding MMS-user for the purpose of coordination of activities with another MMS-user.

13.4.1 Structure

The structure of the component service primitives is shown in Table 60.

Table 60 – DefineSemaphore Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Number of Tokens	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

13.4.1.1 Argument

This parameter shall convey the parameters of the DefineSemaphore service request.

13.4.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name to be associated with the semaphore to be created. It shall be a VMD-specific or Domain-specific name.

13.4.1.1.2 Number of Tokens

This parameter, of type integer, shall specify the number of owners allowed to simultaneously control the semaphore.

13.4.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. A successful result returns no service specific parameters.

13.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.4.2 Service Procedure

The responding MMS-user shall create the semaphore and return the response service primitive. The created Semaphore object shall be initialized as described below:

a) The Semaphore Name attribute shall be initialized to the value of the Semaphore Name parameter.

b) The MMS Deletable attribute shall be initialized to the value true.

c) The Class attribute shall be initialized to the value TOKEN.

d) The Number Of Tokens attribute shall be initialized to the value of the Number Of Tokens parameter.

e) The Number Of Owned Tokens attribute shall be initialized to the value zero (0).

f) The List Of Owner and List Of Requesters attributes shall be initialized to the value empty.

g) An Entry Condition object shall be created and the Entry Condition Reference attribute shall be set too reference this object. The Entry Condition object shall be initialized as follows:

   1) The Event Condition Name attribute shall be set equal to the value of the Semaphore Name attribute of this semaphore.

   2) The MMS Deletable attribute shall be set equal to false.

   3) The Event Condition Class shall be set to NETWORK-TRIGGERED.

   4) The Event Condition State shall be set to DISABLED.

   5) The Event Condition Priority shall be set to normalPriority.

   6) The Event Condition Severity shall be set to normalSeverity.

   7) The List of Event Enrollment References shall be set to empty.

The service shall fail and return an error for any of the following conditions:

a) the responding MMS-user is unable to create the requested object due to lack of local resources;

b) the specified Semaphore Name already exists or cannot be created;

c) the requesting MMS-user does not have sufficient local privilege to request the service.

13.5 DeleteSemaphore Service

The DeleteSemaphore service may be used by an MMS-user in order to delete an MMS deletable semaphore.

13.5.1 Structure

The structure of the component service primitives is shown in Table 61.

Table 61 – DeleteSemaphore Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

13.5.1.1 Argument

This parameter shall convey the parameter of the DeleteSemaphore service request.

13.5.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name of the semaphore to be deleted.

13.5.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. A successful result returns no service specific parameters.

13.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.5.2 Service Procedure

The responding MMS-user shall delete the specified semaphore and the Event Condition Object referenced by the Event Condition Reference attribute of the semaphore, and shall return the positive response service primitive. The service shall fail and return an error for any of the following conditions:

a) The semaphore is unknown, or is not MMS deletable, or cannot be deleted due to local reasons.

b) The semaphore has at least one active owner.

13.6 ReportSemaphoreStatus Service

The ReportSemaphoreStatus service may be used by an MMS-user to obtain the summarized status of a semaphore.

13.6.1 Structure

The structure of the component service primitives is shown in Table 62.

Table 62 – ReportSemaphoreStatus Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Class			M	M(=)	
Number Of Tokens			M	M(=)	
Number Of Owned Tokens			M	M(=)	
Number Of Hung Tokens			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

13.6.1.1 Argument

This parameter shall convey the parameters of the ReportSemaphoreStatus service request.

13.6.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the Token or Pool Semaphore for which the status is to be supplied.

13.6.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated, the following parameters shall be included.

13.6.1.2.1 MMS Deletable

This parameter, of type boolean, shall specify if true that the Semaphore may be deleted using the DeleteSemaphore service.

13.6.1.2.2 Class

This parameter, of type integer, shall specify the class of the semaphore and shall have either the value TOKEN, or POOL.

13.6.1.2.3 Number of Tokens

This parameter, of type integer, shall specify the maximum number of owners allowed by the semaphore.

13.6.1.2.4 Number Of Owned Tokens

This parameter, of type integer, shall specify the current number of owned tokens of the semaphore, whose associated Semaphore Entry is not in the HUNG state.

13.6.1.2.5 Number Of Hung Tokens

This parameter, of type integer, shall specify the number of owned tokens whose associated Semaphore Entry is in the HUNG state.

13.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.6.2 Service Procedure

The responding MMS-user shall determine the status of the specified semaphore and return the response service primitive. The service shall fail and return an error if the Semaphore Name is unknown or cannot be accessed.

13.7 ReportPoolSemaphoreStatus Service

The ReportPoolSemaphoreStatus service may be used by a requesting MMS-user to obtain the name and the state of the Named Tokens controlled by a Pool Semaphore.

13.7.1 ReportPoolSemaphoreStatus Structure

The structure of the component service primitives is shown in Table 63.

Table 63 – ReportPoolSemaphoreStatus Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Name To Start After	U	U(=)			
Result(+)			S	S(=)	
List Of Named Token			M	M(=)	
Free Named Token			S	S(=)	
Owned Named Token			S	S(=)	
Hung Named Token			S	S(=)	
More Follows			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

13.7.1.1 Argument

This parameter shall convey the parameters of the ReportPoolSemaphoreStatus service request.

13.7.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the semaphore for which the List Of Named Token parameter is to be supplied by the responding MMS-user.

13.7.1.1.2 Name To Start After

This optional parameter is present when the requesting MMS-user wishes the list returned by the responding MMS-user to begin with a name other than the first name in the list. If the Name To Start After parameter does not match an existing Named Token in the responding MMS-user, then the collating sequence of the International Reference Version of ISO 646 should be used to determine the name to start after.

13.7.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following response parameters shall be included.

13.7.1.2.1 List Of Named Token

This parameter is a list, possibly empty. Each element of the list shall be one of the following parameters:

13.7.1.2.1.1 Free Named Token

This parameter, of type Identifier, shall contain a Named Token in the state FREE.

13.7.1.2.1.2 Owned Named Token

This parameter, of type Identifier, shall contain a Named Token in the state OWNED, whose associated Semaphore Entry is not in the HUNG state.

13.7.1.2.1.3 Hung Named Token

This parameter, of type Identifier, shall contain a Named Token in the state OWNED, whose associated Semaphore Entry is in the HUNG state.

13.7.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportPoolSemaphoreStatus requests are necessary to retrieve all the requested information. If true, more requests are necessary. If false, then either the List Of Named Token contains the last item in the list, or the List Of Named Token is empty.

13.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.7.2 Service Procedure

The responding MMS-user shall return a List Of Named Token parameter, which is a sublist, possibly empty, ordered by the collating sequence of the International Reference Version of ISO 646, of the List Of Named Token controlled by the specified Pool Semaphore. This List Of Named Token is itself ordered by the same collating sequence. The List Of Named Token returned begins at the beginning of the list if the Name To Start After parameter is not provided in the service indication; otherwise, it begins at the first name, in the order specified by the collating sequence, after the value provided by the Name To Start After parameter.

More Follows shall be set to true if more items remain to be reported after this list is processed; otherwise this parameter shall be set to false. If the List of Named Tokens is empty, More Follows shall be false.

The service shall fail and return an error for any of the following conditions:

a) the Semaphore Name is unknown or cannot be accessed.

b) the Semaphore Name does not reference a Pool Semaphore.

13.8 ReportSemaphoreEntryStatus Service

The ReportSemaphoreEntryStatus service may be used by an MMS-user to obtain the detailed status of the semaphore entries dependent on a semaphore.

13.8.1 Structure

The structure of the component service primitives is shown in Table 64.

Table 64 – ReportSemaphoreEntryStatus Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
State	M	M(=)			
Entry ID To Start After	U	U(=)			
Result(+)			S	S(=)	
List Of Semaphore Entry			M	M(=)	
Entry ID			M	M(=)	
Entry Class			M	M(=)	
Application Reference			M	M(=)	
Named Token			C	C(=)	
Priority			M	M(=)	
Remaining Time Out			C	C(=)	
Abort On Time Out			C	C(=)	
Relinquish If Connection Lost			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.8.1.1 Argument

This parameter shall convey the parameters of the ReportSemaphoreEntryStatus service request.

13.8.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the semaphore for which the List Of Semaphore Entry is to be supplied.

13.8.1.1.2 State

This parameter shall specify the Entry State attribute of the Semaphore Entries whose status is to be reported. This parameter may have the value QUEUED, OWNER or HUNG.

13.8.1.1.3 Entry ID To Start After

This optional parameter, of type octetstring, shall specify that the requesting MMS-user wants to be returned only the sublist of Semaphore Entry beginning after the SemaphoreEntry whose Entry ID is provided with the parameter.

13.8.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following response parameters shall be included.

13.8.1.2.1 List Of Semaphore Entry

This parameter shall be a list, possibly empty, of attributes of Semaphore Entry objects matching the parameters specified by the requesting MMS-user. Each element of the list shall contain the following parameters:

13.8.1.2.1.1 Entry ID

This parameter, of type octetstring, shall identify the Semaphore Entry represented by this element of the list.

13.8.1.2.1.2 Entry Class

This parameter shall specify the class of the Semaphore Entry and may have the value SIMPLE or MODIFIER.

13.8.1.2.1.3 Application Reference

This parameter, of type ApplicationReference, shall identify the MMS-user which created the SemaphoreEntry.

13.8.1.2.1.4 Named Token

This parameter, of type Identifier, shall be returned if the Named Token attribute is defined in the Semaphore Entry. It shall identify the optional required Named Token or the owned Named Token of a Pool Semaphore, depending upon the state of the Semaphore Entry.

13.8.1.2.1.5 Priority

This parameter, of type integer, shall contain the Priority provided at the creation of the Semaphore Entry.

13.8.1.2.1.6 Remaining Time Out

This optional parameter, of type integer, shall contain either the Remaining Acquisition Delay in milliseconds if the Semaphore Entry is in the state QUEUED, or the remaining Control Time Out in milliseconds if it is in the state OWNER. This parameter shall not be provided if no time out was specified at the creation of the Semaphore Entry.

13.8.1.2.1.7 Abort On Time Out

This parameter, of type boolean, shall contain the value specified at the creation of the Semaphore Entry. This parameter shall not be provided if the Control Time Out parameter was not specified at the creation of the Semaphore Entry.

13.8.1.2.1.8 Relinquish If Connection Lost

This parameter, of type boolean, shall contain the value specified at the creation of the Semaphore Entry.

13.8.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportSemaphoreEntryStatus requests are necessary to retrieve all the information. If true, more requests are necessary. If false, then either the List Of Semaphore Entry parameter contains the last item in the list, or it is empty.

13.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

13.8.2 Service Procedure

The responding MMS-user shall maintain three different logical lists of Semaphore Entries for each semaphore, one for each state QUEUED, OWNER, HUNG. Each list shall be ordered by a locally defined algorithm. The responding MMS-user shall return a List Of Semaphore Entry parameter, which is a list, possibly empty, containing the ordered items matching the parameters provided in the service request. If the service request does not provide an Entry ID To Start After parameter, the returned list shall begin at the beginning of the list maintained by the VMD. Otherwise, it shall begin after the item provided by the parameter. If the list to be returned is too large to be sent in a single service response, the VMD shall return the first item of the list and the More Follows parameter with the value true; the requesting MMS-user may send another service request, with the Entry ID To Start After parameter containing the last Entry ID returned. If the Entry ID specified by the Entry ID To Start After parameter does not reference an item in the requested list, the VMD shall return an error.

NOTE – Due to the dynamics of Semaphore Entry lists, the range and the state of a specific item of the list may change between two service requests and the application should be aware of this. This service is designed for handling recovery procedures in the case of deadlocks or connection aborts, and such situations are static.

The service shall fail and return an error for any of the following conditions:

a) the Semaphore Name is unknown or cannot be accessed.

b) the Entry ID does not reference an item of the specified list.

13.9 AttachToSemaphore Modifier

The AttachToSemaphore Modifier is provided so that the processing of a service may be delayed at a responding MMS-user until the control of a semaphore has been granted by this service. Services that are modified are not acted on immediately, but are placed in a queue corresponding to a semaphore at the responding MMS-user; when this service request rises to the top of the queue, it is acted on.

13.9.1 Structure

The structure of the component service primitives is shown in Table 65.

Table 65 – AttachToSemaphore Modifier

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Attach To Semaphore	S	S(=)			
Semaphore Name	M	M(=)			
Named Token	C	C(=)			
Priority	M	M(=)			
Acceptable Delay	U	U(=)			
Control Time Out	U	U(=)			
Abort On Time Out	C	C(=)			
Relinquish If Connection Lost	M	M(=)			

13.9.1.1 Attach To Semaphore

When a VMD implements the Attach To Semaphore service modifier, the Attach To Semaphore parameter is provided as a parameter of the List of Modified parameter for each confirmed service request (see 5.4). The sub-parameters of the Attach To Semaphore modifier are specified as follows:

13.9.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the semaphore under which the modified service request is to be controlled.

13.9.1.1.2 Named Token

This parameter, of type Identifier, shall be provided if the semaphore is a Pool Semaphore and shall not be provided otherwise. It shall specify which Named Token controlled by the Pool Semaphore the MMS-user wishes to be granted control. The dynamic allocation of Named Token by the responding MMS-user is not allowed, since the modified request could not identify the allocated Named Token.

13.9.1.1.3 Priority

This parameter, of type integer, shall identify the priority that this modified service request should have as compared to (possible) other TakeControl requests, as well as to services using the AttachToSemaphore Modifier, which are awaiting the same semaphore.

13.9.1.1.4 Acceptable Delay

This optional parameter, of type integer, shall indicate the duration of time in milliseconds for which the requesting user is willing to wait for control to be allocated. If the time out expires, the service to which the modifier is attached shall fail and a error shall be returned. If a zero value is specified for acceptable delay, this means that no delay is acceptable. (That is, if the semaphore or Named Token is not immediately available, the service to which the Modifier is attached fails.) If no value is specified for acceptable delay, this shall be interpreted as meaning that any delay is acceptable (that is "wait forever"). The granularity of one millisecond is not required. The granularity supported by the VMD shall be specified in the static conformance statement of the PICS.

13.9.1.1.5 Control Time Out

This optional parameter, of type integer, shall specify the duration of time in milliseconds for which control of the semaphore may be held (after it is obtained). If this time limit is exceeded and the value of the Abort On Time Out parameter is true, the Application Association shall be aborted by a provider abort, and the Semaphore Entry shall either be released or placed in the HUNG state according to the Relinquish if Connection Lost parameter. If this time limit is exceeded and the value of the Abort On Time Out parameter is false, the Semaphore Entry shall be placed in the HUNG state and the related Event Condition shall be signalled. If no value is provided, no control time out applies, and the semaphore may be held indefinitely. The granularity of one millisecond is not required. The granularity supported by the VMD shall be specified in the static conformance statement of the PICS.

13.9.1.1.6 Abort On Time Out

This boolean parameter shall be provided if the Control Time Out parameter is provided. The value true means that the Application Association shall be aborted if the Control Time Out expires. The value false shall mean that the Application Association is to be maintained and the related Event Condition signalled if the Control Time Out expires.

13.9.1.1.7 Relinquish If Connection Lost

This boolean parameter shall specify if true that the responding MMS-user shall relinquish the semaphore if the owner of the semaphore loses the ability to control it, either due to the loss of the association used for acquiring the semaphore, or to a local failure. The value false shall mean that in the same situation the responding MMS-user shall place the Semaphore Entry in the HUNG state.

13.9.2 Service Procedure

When the responding MMS-user processes this modifier as part of the transaction object processing of the modified service, it shall create a Semaphore Entry object and add it to the queue of the semaphore. The attributes of the created Semaphore Entry shall be initialized as follows:

a) The Entry ID attribute shall contain a locally defined value which shall be unique for each semaphore.

b) The Entry Class attribute shall be initialized to MODIFIER.

c) The Semaphore Name attribute shall be initialized to the value of the Semaphore Name parameter.

d) The Requester Application Reference shall be initialized to the value identifying the Application Process issuing the service.

e) The Application Association Local Tag attribute shall be initialized by a value identifying the application association.

f) The Invoke ID attribute shall be initialized to the value of the Invoke ID parameter of the indication. This attribute references the Transaction Object that handles the possible nesting of modifiers.

g) The Named Token attribute shall be initialized to the value of the Named Token parameter of the indication if provided, otherwise to the value UNDEFINED. If the semaphore is a Pool Semaphore, this parameter shall be provided.

h) The Priority attribute shall be initialized to the value of the Priority parameter.

i) If no value of the Acceptable Delay parameter is provided, then the value of the Remaining Acquisition Delay attribute shall be set to the value FOREVER. If a value is provided, then the Acquisition Delay attribute shall be initialized to the value of the Acceptable Delay parameter and the associated timer activated.

j) If no value of the Control Time Out parameter is provided, then the value of the Remaining Control Time Out attribute shall be set to the value FOREVER. If a value is provided, then the Remaining Control Time Out attribute shall be initialized to the value of the Control Time Out parameter and the associated timer activated.

k) The Abort On Time Out attribute shall be initialized to the value of the Abort On Time Out parameter.

l) The Relinquish If Connection Lost attribute shall be initialized to the value of the Relinquish If Connection Lost parameter.

m) The Entry State shall be initialized to the value QUEUED.

Upon acquisition of the semaphore, the request shall be released for further processing, under control of the Transaction Object. Upon completion of all processing associated with the modified service, the semaphore shall be relinquished. The semaphore shall also be relinquished (1) upon any abnormal completion of the request such as a Cancel service, or (2) by an abort of the association if the Relinquish If Connection Lost parameter so specifies.

The service modified by an AttachToSemaphore Modifier may fail and return an error for any of the following conditions:

a) The Semaphore Name or the Named Token specified in the service request is unknown or cannot be accessed.

b) The specified delay has expired, the service request has been cancelled by a Cancel service, a time out has occurred, as a result of local action, or for any other reason before the control of the semaphore is effective.

c) The VMD cannot process the service request due to a lack of local resources, for example the semaphore queues are full.

d) Any of the errors which are applicable to the service instance without modification.

13.10 Conformance

The Semaphore Management Services define parameter conformance requirements for the VMD. These requirements are described below.

13.10.1 Support for Time

The Protocol Implementation Conformance Statement (PICS) shall provide the granularity of time supported in the processing of the Remaining Acquisition Delay and Remaining Control Time Out attributes of a Semaphore Entry object.

13.10.2 Support for Priority

The Protocol Implementation Conformance Statement (PICS) shall specify the algorithm executed in the processing of the Priority attribute of a Semaphore Entry object.

14 Operator Communication Services

The Operator Communication Services provide a mechanism for communicating with an Operator Station which allows display of data, entry of data, or both. The Operator Communication Services are:

Input;

Output.

NOTE – The MMS Operator Communication Services are intended to be restrictive and simple. In the OSI environment, general virtual terminal capabilities, including extensive display management, are provided through the ISO Virtual Terminal (VT) service and protocol, defined in ISO 9040 and ISO 9041, respectively. Systems requiring a general operator communication facility should make use of VT services instead of the MMS Operator Communication Services.

14.1 The Operator Communications Model

The Operator Communication Services uses an Operator Station object to describe how the Input and Output services are used. MMS does not specify a flow control mechanism to manage the input and output function for these objects. It is the responsibility of implementation to provide locally defined flow control mechanisms to ensure data integrity on the Operator Station for multiple transactions for a single MMS-user.

NOTE – When more than one MMS-user is competing for the operator station object, the MMS-user should use the MMS Semaphore Management services to manage the control of the operator station object.

14.1.1 The Operator Station Object

Object: Operator Station

```
Key Attribute: Operator Station Name
Attribute: Station Type (ENTRY, DISPLAY, ENTRY-DISPLAY)
Constraint: Station Type = ENTRY
        Attribute: Input Buffer
        Attribute: State (IDLE, WAITING-FOR-INPUT-STRING,
                          INPUT-BUFFER-FILLED)
Constraint: Station Type = DISPLAY
        Attribute: List Of Output Buffer
        Attribute: State (IDLE, OUTPUT-BUFFERS-FILLED)
Constraint: Station Type = ENTRY-DISPLAY
        Attribute: Input Buffer
        Attribute: List Of Output Buffer
        Attribute: State (IDLE, DISPLAY-LIST-OF-PROMPT-DATA,
                          WAITING-FOR-INPUT-STRING, INPUT-BUFFER-FILLED,
                          OUTPUT-BUFFERS-FILLED)
Attribute: Additional Detail
```

Operator Station Name

The Operator Station Name attribute uniquely identifies the Operator Station object. An Operator Station Name is of type Identifier.

Station Type

The Station Type attribute indicates the type of station for the Operator Station object. There are three types of Operator Station objects: ENTRY, DISPLAY, and ENTRY-DISPLAY.

Input Buffer

The Input Buffer attribute contains the value of the Input String parameter of the Input service response. Operator Station objects of type ENTRY or ENTRY-DISPLAY shall have this attribute.

List Of Output Buffer

The List Of Output Buffer attribute contains either the value of the List Of Output Data parameter for Output service request, or the value of the List Of Prompt Data parameter for the Input service request. Only Operator Station objects of type DISPLAY or ENTRY-DISPLAY shall have this attribute.

State

The State attribute contains the state of the Operator Station object. An Operator Station object of type ENTRY may be in one of three states: IDLE, WAITING-FOR-INPUT-STRING, and INPUT-BUFFER-FILLED. An Operator Station object of type DISPLAY may be in one of two states: IDLE and OUTPUT-BUFFERS-FILLED. An Operator Station object of type ENTRY-DISPLAY may be in one of five states: IDLE, DISPLAY-LIST-OF-PROMPT-DATA, WAITING-FOR-INPUT-STRING, INPUT-BUFFER-FILLED, and OUTPUT-BUFFERS-FILLED. In the IDLE state, the object may accept Input or

Output service requests, depending on its type. In any other state, the operator station object is busy executing a service request and may be unavailable for additional service requests depending upon the local flow control mechanism used by the implementation.

Additional Detail

This attribute is included to allow for additional attributes to be added by the Companion Standards.

14.1.1.1 Types of Operator Station Objects

There are three types of Operator Station objects, which differ in their ability to accept Input service requests, Output service requests, or both. These three types are summarized as follows:

a) An Operator Station object of type ENTRY can accept Input service requests only. No prompt data is allowed with this service.

b) An Operator Station object of type DISPLAY can accept Output service requests only.

c) An Operator Station object of type ENTRY-DISPLAY can accept both Input and Output service requests. The List Of Prompt Data parameter is allowed on the Input service request. Furthermore, data entered for the Input service response may be echoed on to the display of the Operator Station.

14.1.2 Relationship between the MMS Object and the Physical Device

The relationship between the physical operator station device and the Input Buffer attribute and the List Of Output Buffer attribute of the operator station object is specified by a pair of abstract functions which work with physical data storage areas. The Input Buffer attribute of the MMS operator station object is associated with an input buffer, which is a physical storage area containing the value of the Input String which is entered by the operator. The List Of Output Buffer attribute of the MMS operator station object is associated with a set of output buffers, which is a set of physical storage areas containing the values of either the List Of Prompt Data parameter from the Input service request, or the List Of Output Data parameter from the Output service request. The abstract functions which provide the mapping of the values contained in these buffers to the physical device in a way which allows the operator to interact with the Input and Output service requests are described below.

14.1.2.1 The D-Put Function

The D-Put function obtains the values contained in the set of output buffers and writes these values to the physical display of the operator station device. This occurs when the List Of Output Data parameter values need to be displayed for the Output service request, or when the List Of Prompt Data parameter values need to be displayed for the Input service request.

14.1.2.2 The E-Get Function

The E-Get function obtains the Input String parameter value which is entered by the operator and writes it to the input buffer. This function completes its process when a locally defined end-of-input indication is received.

14.1.3 Operator Station State Diagram

The state diagram shown in Figure 12 shows the state transitions possible for an Operator Station object of type ENTRY-DISPLAY. If the Operator Station object is of type ENTRY, then state transitions 2 and 3 shall not be permitted. If the Operator Station object is of type DISPLAY, the state transitions 1, 4, 5, and 6 shall not be permitted.

Transitions of the Operator Station state diagram are as follows:

1 - Output.indication

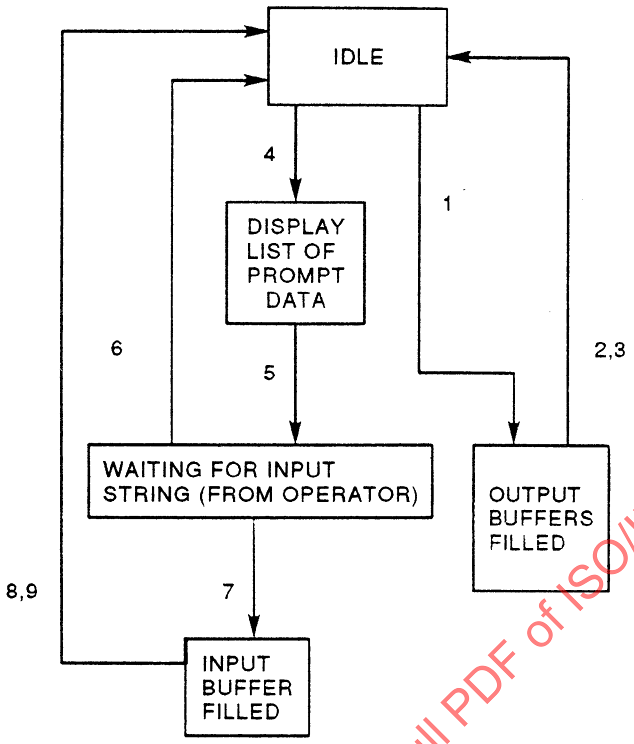

IDLE
4
1
DISPLAY LIST OF PROMPT DATA
6
5
2,3
WAITING FOR INPUT STRING (FROM OPERATOR)
OUTPUT BUFFERS FILLED
8,9
7
INPUT BUFFER FILLED

Figure 12 – Operator Station State Diagram

2 - Output.response (+)
3 - Output.response (-)
4 - Input.indication
5 - (D-Put function finished displaying List Of Prompt Data, if any. If present, Input Time Out begins.)
6 - Input.response (-) due to a time out
7 - (E-Get function finished entering the Input String into input buffer. If present, Input Time Out stops.)
8 - Input.response (+)
9 - Input.response (-)

14.2 Input Service

An MMS-user may request data from an Operator Station object of type ENTRY or ENTRY-DISPLAY via the Input service. If the Input service request identifies an Operator Station object of type ENTRY-DISPLAY, then the MMS-user has the option of allowing prompt data to be displayed prior to the input of data via the List Of Prompt Data parameter, and has the option of echoing the Input String parameter to the display of the station.

14.2.1 Structure

The structure of the component service primitives is shown in Table 66.

Table 66 – Input Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Operator Station Name		M	M(=)			
Echo		M	M(=)			
List Of Prompt Data		U	U(=)			
Input Time Out		U	U(=)			
Result(+)	(COMP)			S	S(=)	
Input String				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

14.2.1.1 Argument

This parameter shall convey the service specific parameters of the Input service request.

14.2.1.1.1 Operator Station Name

This parameter, of type Identifier, shall identify the Operator Station object from which the Input String is requested. The Operator Station Name provides the value of the Operator Station Name attribute of the Operator Station object.

14.2.1.1.2 Echo

This parameter, of type boolean, shall indicate whether (true) or not (false) the value of the Input String parameter is to be displayed on the Operator Station. When the Echo parameter of the Input service request is equal to true, the operator station object type is ENTRY-DISPLAY, and the Output service CBB is supported, the value contained in the input buffer shall be copied into the output buffers and is displayed by the D-Put function. These actions shall not occur if the Echo parameter is false. This parameter shall be ignored for Operator Station objects of type ENTRY.

14.2.1.1.3 List Of Prompt Data

This optional parameter shall be a list of character strings. This parameter shall only be present if the Operator Station object type is ENTRY-DISPLAY and the Output service CBB is supported. The D-Put function shall display the value of the List Of Prompt Data contained in the output buffers prior to the invocation of the E-Get function which obtains the information provided by the operator and writes it into the input buffer. Each element of the list shall represent a single line on the display.

14.2.1.1.4 Input Time Out

This optional parameter, of type integer, shall specify a time out period (in seconds) for an operator to complete the process of entering the value of the Input String on the Operator Station. The VMD shall begin timing this Input service request immediately after the processing of the service request of the transaction object has begun, or, if the List Of Prompt Data parameter is present, immediately after the D-Put function has finished writing the value of this parameter to the display of the Operator Station. Timing shall stop for this Input service request immediately after the E-Get function has written the Input String entered by the operator to the input buffer.

The values for the Input Time Out parameter ranges from 0 to 2**31-1. If this parameter is present and the specified time out period expires, an error result shall be returned. If this parameter is absent, then an unlimited time out period is indicated. A zero value for this parameter shall result in an Error Response, except when the E-Get function has indicated that it has already written the Input String entered by the operator into the input buffer.

14.2.1.2 Result(+)

The Result(+) parameter shall indicate the service request has succeeded. When success is indicated, the Input String parameter shall also be included.

14.2.1.2.1 Input String

This parameter, of type character string, is the data provided by the operator. The Input String parameter in the Input service response shall contain the value of one input buffer. If multiple strings of input are required, multiple Input service requests shall be issued. The determination of the end of a line of Input String is a local issue, but the end of line indication shall not be included in the Input String.

14.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

14.2.2 Service Procedure

If a prompt has been specified, each element in the List Of Prompt Data parameter shall be displayed by the D-Put function on the display of the Operator Station object specified at the responding MMS-user. An Input String shall be entered by the operator, which is then returned to the requesting MMS-user. If an input time out is specified, the Input String shall be entered by the operator and copied into the input buffer by the E-Get function before the expiration of the input time out. If no input time out is specified, an unlimited amount of time is available to enter the Input String.

The service shall fail and return an error for any of the following conditions:

a) no Operator Station object with the Operator Station Name exists,

b) the Input service request has timed out,

c) the Operator Station object is not in the IDLE state,

d) the Operator Station object cannot support the Input Time Out parameter specified in the Input service request,

e) the Operator Station object cannot support the List Of Prompt Data parameter because the Constraint Attribute is not ENTRY-DISPLAY.

14.3 Output Service

An MMS-user may request the Output service to display data on an Operator Station object.

14.3.1 Structure

The structure of the component service primitives is shown in Table 67.

Table 67 – Output Service

Table 67 (Cont.) – Output Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Operator Station Name		M	M(=)			
List Of Output Data		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

14.3.1.1 Argument

This parameter shall convey the service specific parameters of the Output service request.

14.3.1.1.1 Operator Station Name

This parameter, of type Identifier, shall identify the Operator Station object to which output data is to be displayed. The Operator Station Name provides the value of the Operator Station Name attribute of the Operator Station object.

14.3.1.1.2 List Of Output Data

This parameter shall be a list of character strings. Each character string shall be displayed on the display of the Operator Station object by the D-Put function. Each element of the list represents a single line on the display.

14.3.1.2 Result(+)

The Result(+) parameter shall indicate the service request has succeeded. A successful result returns no service specific parameters.

14.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

14.3.2 Service Procedure

The values supplied in the List Of Output Data parameter shall be displayed on the display of the Operator Station object referenced by the Operator Station Name parameter.

The service shall fail and an error response shall result if no Operator Station object with the Operator Station Name exists.

15 Event Management Services

The event management services provide facilities which allow a client MMS-user to define and manage event objects at a VMD and to obtain notifications of event occurrences. These facilities are provided through extension of the VMD model to include three (3) additional objects and by providing a set of nineteen (19) services and one (1) service modifier to operate upon these objects. Subclause 15.1 describes the event management model. This is followed, in 15.2 through 15.21, by a description of the event management services and service modifier. The state tables associated with the objects of the event management model are described in 15.22. Finally, 15.23 describes conformance requirements unique to event management.

15.1 The Event Management Model

The event management services add three objects to the VMD model and provide services which operate on these objects. The objects which are defined by this clause are the Event Condition object, the Event Action object, and the Event Enrollment object. Each of these objects models a specific aspect of the state information associated with the management of MMS Events. Only those events which have the potential of causing the initiation of an EventNotification service request are considered by this model.

The Event Condition object models that portion of the state information which is concerned with event detection and prioritization. It also includes information which assists in the determination of active "alarms".

The Event Action object models that portion of the state information which is concerned with the execution of MMS services upon the occurrence of an event.

The Event Enrollment object is used optionally to tie a given Event Condition object to a given Event Action object and to tie the event notifications resulting from an event transition to a client and thus, an application association. The Event Enrollment object additionally includes state information which may be used for tracking and coordinating client responses to alarm event notifications. The Event Enrollment object may also be employed for the delayed (conditional) execution of any confirmed service through the use of the Attach To Event Condition Modifier.

The interrelationship between these three objects is illustrated in Figure 13.

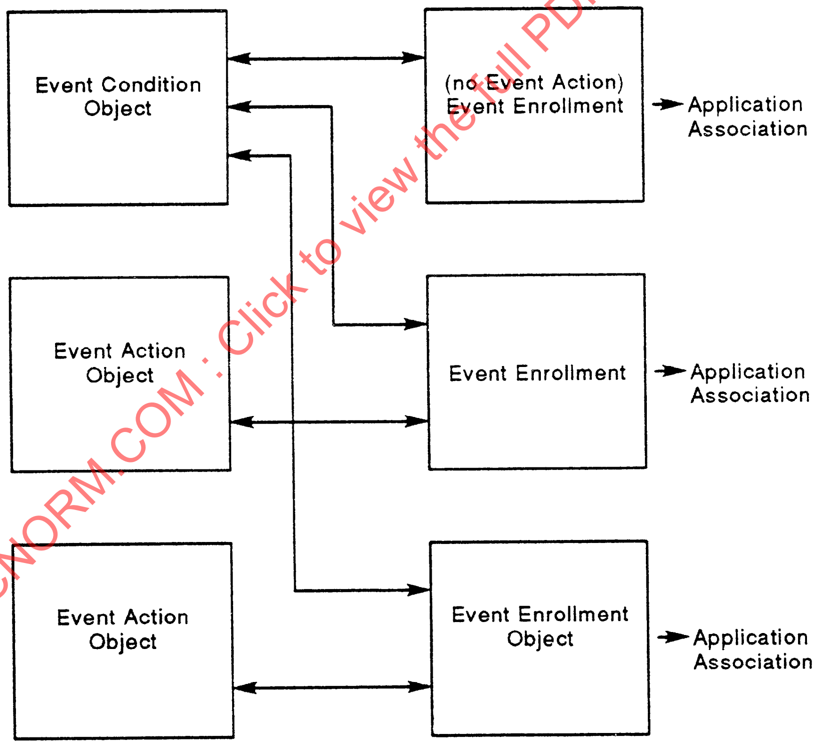


Figure 13 – Relationship Between Event Management Objects

Additional details on the Event Condition, Event Action and Event Enrollment objects, and the services and modifier which operate upon these objects, are provided below.

15.1.1 The Event Condition Object

The Event Condition object models the MMS-visible aspects of an Event Condition. The event management model defines two classes of Event Condition objects. They are the network-triggered Event Condition object and the monitored Event Condition object.

A network-triggered Event Condition object models an event occurring due to the explicit request of a client using the TriggerEvent service. The network-triggered Event Condition object also models events which occur internally, as a result of autonomous VMD action.

A monitored Event Condition object is a virtual representation of an aspect of VMD activity which, due to application design criteria, represents a significant occurrence in the processing of the VMD. Events associated with a monitored Event Condition object are detected by the autonomous action of the VMD.

Either class of Event Condition object includes the potential for VMD initiation of EventNotification service executions, through the Procedure for Event Transition Processing (see 15.1.4.2.) The attributes of an Event Condition object are specified below, followed by a description of the services which operate on the Event Condition object.

15.1.1.1 Attributes of an Event Condition Object

Object: Event Condition

```
Key Attribute: Event Condition Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Event Condition Class (NETWORK-TRIGGERED, MONITORED)
Attribute: State (DISABLED, IDLE, ACTIVE)
Attribute: Priority
Attribute: Severity
Attribute: Additional Detail
Attribute: List Of Event Enrollment Reference
Constraint: Event Condition Class = MONITORED
    Attribute: Enabled (TRUE, FALSE)
    Attribute: Alarm Summary Reports (TRUE, FALSE)
    Attribute: Monitored Variable Reference
    Attribute: Evaluation Interval
    Attribute: Time Of Last Transition To Active
    Attribute: Time Of Last Transition To Idle
```

Event Condition Name

The Event Condition Name attribute shall uniquely identify the Event Condition object. An Event Condition Name is an MMS Object Name. It may have VMD-specific, Domain-specific or AA-specific scope.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) this Event Condition object may be deleted using the DeleteEventCondition service.

Event Condition Class

This attribute shall indicate the class of the Event Condition object. This attribute shall contain the value NETWORK-TRIGGERED, or the value MONITORED, depending on whether the Event Condition object is a network-triggered Event Condition object or a monitored Event Condition object.

State

The State attribute shall maintain the current state of the Event Condition object. The value of this attribute is always DISABLED for a network-triggered Event Condition object.

The value of the State attribute of a monitored Event Condition object shall be determined by the value of the Enabled attribute, by the value of the Monitored Variable Reference attribute, and by the value of the "referenced variable," or the underlying VMD-variable, the value of which is determinable through the use of the Variable object referenced by the value of the Monitored Variable Reference attribute and the V-Get function (see 12.1.1.2.1.) The value of the State attribute shall be DISABLED if the value of the Enabled attribute is false. If the value of the Enabled attribute is true and the Monitored Variable Reference attribute does not have the value UNSPECIFIED or UNDEFINED, then the value of the State attribute shall be:

a) IDLE if the value of the referenced variable is determined to be false.

b) ACTIVE if the value of the referenced variable is determined to be true.

The value of the State attribute shall be IDLE or ACTIVE as determined by local means if the Enabled attribute has the value true and the value of the Monitored Variable Reference attribute has the value UNSPECIFIED or UNDEFINED.

Priority

The Priority attribute shall indicate the importance of the Event Condition object relative to other Event Condition objects defined at the VMD. This attribute shall rank the MMS defined Event Condition objects at a VMD for the purpose of evaluating their state. It shall additionally rank all Event Condition objects for the purpose of executing the Procedures for Event Transition Processing (see 15.1.4.2.) The Priority attribute shall be an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero shall represent the highest priority. One hundred twenty seven (127) shall represent the lowest priority. By convention, sixty four (64) shall represent the "normal" priority. Behaviour of the VMD with respect to the influence of the value of the priority attribute shall be a local issue.

Severity

The Severity attribute shall represent the effect of the event on the process which is being controlled. The Severity attribute shall be an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero shall indicate the most severe and one hundred twenty seven shall indicate the least severe. By convention, sixty four (64) shall represent the "normal" severity. Behaviour of the VMD with respect to the influence of the value of the severity attribute shall be a local issue.

Additional Detail

The Additional Detail attribute shall maintain application, or device-class specific state information. The semantics for this attribute shall be specified by a Companion Standard. If the abstract syntax defined in ISO/IEC 9506-2, clause 19, has been used for definition of the Event Condition object, then this attribute shall not exist.

List Of Event Enrollment Reference

The List Of Event Enrollment Reference attribute shall contain a list of references to zero or more Event Enrollment objects, each of which specifies a reference to the Event Condition object in the Event Condition Reference attribute of the Event Enrollment object.

Enabled

The Enabled attribute exists for monitored Event Conditions only. It shall specify whether (true) or not (false) changes in the determined value of the Event Condition object's referenced variable shall cause invocation of the Procedure for Event Transition Processing (see 15.1.4.2) for the Event Enrollment objects referenced in the List of Event Enrollment Reference attribute.

Alarm Summary Reports

The Alarm Summary Reports attribute shall exist for monitored Event Condition objects only. If true, it shall specify that the Event Condition object shall be considered for inclusion in responses to the GetAlarmSummary service without regard to the value of the State attribute of the Event Condition object, and the value of the Alarm Acknowledgement Rule attributes of any referenced Event Enrollment objects. If false, the Event Condition object shall not be considered for inclusion in alarm summaries unless at least one referenced Event Enrollment object contains a value for the Alarm Acknowledgement Rule attribute which is not equal to NONE.

Monitored Variable Reference

The Monitored Variable Reference attribute shall exist for monitored Event Conditions only. For a monitored Event Condition object, it shall reference a Variable object (Unnamed or Named) of type boolean. The value determinable through the use of the referenced Variable object is monitored (by the VMD through the V-Get function) to determine the value of the State attribute of an enabled monitored Event Condition object according to the rules specified for the State attribute.

NOTE – The relationship between the value determined through the use of the referenced monitored variable object and the state of the process controlled by the VMD is a local issue.

For a locally defined Event Condition, or for an Event Condition created through the use of the CreateProgramInvocation service, this attribute shall have the value UNSPECIFIED, indicating that the event transitions are not determined by the value of an MMS-visible object.

If the referenced monitored variable is deleted, for example, through the deletion of a Domain, or loss of an application association, the value of the Monitored Variable Reference attribute of the Event Condition object shall be set to UNDEFINED. In this case, the Procedure for Event Transition Processing (see 15.1.4.2) shall never be invoked.

Evaluation Interval

The Evaluation Interval attribute shall exist for a monitored Event Condition only. The Evaluation Interval attribute shall specify the maximum acceptable time, in milliseconds, between determinations of the value of the Event Condition object's State attribute.

Time Of Last Transition To Active

The Time Of Last Transition To Active attribute shall only exist for a monitored Event Condition object. It shall record the time (date and time of day or Time Sequence Identifier) of the last detected transition of the value of the Event Condition object's state to ACTIVE. If the Event Condition object's State attribute has never had the value ACTIVE, then this attribute shall have the value UNDEFINED.

This attribute shall not be included in reports unless the Alarm Summary Reports attribute contains the value true, or the Event Condition object has been referenced by an Event Enrollment object which has the value of the Alarm Acknowledgement Rule attribute not equal to NONE.

Time Of Last Transition To Idle

The Time Of Last Transition To Idle attribute shall exist only for a monitored Event Condition object. It shall record the time (date and time of day or Time Sequence Identifier) of the last detected transition of the Event Condition object's state attribute to the value IDLE. If the Event Condition object's State attribute has never had the value IDLE, then this attribute shall have the value UNDEFINED.

This attribute shall not be included in reports unless the Alarm Summary Reports attribute contains the value true, or the Event Condition object has been referenced by an Event Enrollment object which has the value of the Alarm Acknowledgement Rule attribute not equal to NONE.

15.1.1.2 Operations on Event Condition Objects

The services and modifier which operate upon Event Condition objects are listed below.

a) DefineEventCondition - This service causes the creation of an Event Condition object at a VMD.

b) DeleteEventCondition - This service causes the deletion of one or more Event Condition objects defined at a VMD.

c) GetEventConditionAttributes - This service obtains the values of the descriptive attributes of an Event Condition object defined at a VMD.

d) ReportEventConditionStatus - This service obtains the values of the status attributes of an Event Condition object defined at a VMD.

e) AlterEventConditionMonitoring - This service is used to alter the values of any combination of a monitored Event Condition object's Priority, Enabled, and Alarm Summary Reports attributes.

f) TriggerEvent - This service is used to trigger an event associated with a network-triggered Event Condition object. A replacement for the value of the network-triggered Event Condition object's Priority attribute may be supplied in the service.

g) GetEventEnrollmentAttributes - This service returns the values of the descriptive attributes of Event Enrollment objects referenced by an Event Condition object's List Of Event Enrollment Reference attribute.

h) DefineEventEnrollment - This service creates a new notification Event Enrollment object. This service also updates an Event Condition object's List Of Event Enrollment Reference attribute, by adding a reference to the resultant Event Enrollment object to the list.

i) DeleteEventEnrollment - This service deletes one or more notification Event Enrollment objects, and in addition updates one or more Event Condition object's List Of Event Enrollment Reference attribute by removing references to those Event Enrollment objects which were deleted.

j) AcknowledgeEventNotification - This service may trigger a network-triggered Event Condition object, when the specified object is used as an Acknowledgement Event Condition object for group acknowledgements of Event Notifications.

k) GetAlarmSummary - This service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current status of monitored Event Condition objects and related attributes of their referenced notification Event Enrollment objects.

l) GetAlarmEnrollmentSummary - This service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current alarm status of notification Event Enrollment objects and related attributes of their referenced monitored Event Condition objects.

m) Attach To Event Condition - This service modifier causes execution of a requested confirmed service to be delayed subject to the occurrence of a specified transition of an Event Condition object. A modifier Event Enrollment object is created as a result of execution of this service. This service updates an Event Condition object's List Of Event Enrollment Reference attribute by adding a reference to the resultant modifier Event Enrollment object.

15.1.2 The Event Action Object

An Event Action is an MMS confirmed service which shall be executed whenever a specified transition of an Event Condition object's State attribute is detected. An Event Action object shall represent a requirement for execution of an Event Action as a component of the Procedure for Event Transition Processing (see 15.1.4.2.) The attributes of an Event Action object are described below, followed by a brief description of the services which operate on the Event Action object.

15.1.2.1 Attributes of Event Action objects

Object: Event Action

```
Key Attribute: Event Action Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Confirmed Service Request
Attribute: List Of Modifier
Attribute: List Of Event Enrollment Reference
Attribute: Additional Detail
```

Event Action Name

The Event Action Name attribute shall uniquely identify the Event Action object. An Event Action Name shall be an MMS Object Name. It shall have VMD-specific, Domain-specific or AA-specific scope.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) this Event Action object may be deleted using the DeleteEventAction service.

Confirmed Service Request

The Confirmed Service Request attribute shall identify the service procedure which shall be executed as an Event Action by the VMD during the Procedure for Event Transition Processing (see 15.1.4.2.) This attribute shall additionally contain the service argument to be used when executing the requested service procedure. The result of executing the requested service procedure with the argument provided in this attribute shall be included in the EventNotification service primitive, which shall be generated during the Procedure for Event Transition Processing.

List Of Modifier

The List Of Modifier attribute shall contain the list of modifiers, if any, which shall apply to every execution of the Event Action.

List Of Event Enrollment Reference

The List Of Event Enrollment Reference attribute shall maintain a list of references to Event Enrollment objects which are currently referencing this Event Action object. Zero or more Event Enrollment objects may reference an Event Action object.

Additional Detail

The Additional Detail attribute shall maintain application, or device-class specific state information. The semantics for this attribute shall be specified by a Companion Standard. If the abstract syntax defined in ISO/IEC 9506-2, clause 19, has been used for definition of the Event Action object, then this attribute shall not exist.

15.1.2.2 Operations on Event Actions

The services which operate upon Event Action objects are listed below.

a) DefineEventAction - This service causes the creation of an Event Action object at the VMD.

b) DeleteEventAction - This service causes the deletion of one or more Event Action objects at the VMD.

c) GetEventActionAttributes - This service obtains the values of the descriptive attributes of an Event Action object from a VMD.

d) ReportEventActionStatus - This service obtains the values of the status attributes of an Event Action object from a VMD.

e) DefineEventEnrollment - This service creates a notification Event Enrollment object. This service also updates an Event Action object's List Of Event Enrollment Reference attribute, by adding a reference to the resultant Event Enrollment object to the list.

f) DeleteEventEnrollment - This service deletes one or more notification Event Enrollment objects, and in addition updates one or more Event Action Object's List Of Event Enrollment Reference attribute by removing references to those Event Enrollment objects which were deleted.

15.1.3 The Event Enrollment Object

An Event Enrollment shall represent a request from a client MMS-user to be notified of the occurrence of one or more specified transitions of the State attribute of an Event Condition object, or to delay execution of a confirmed MMS-service until the occurrence of one or more specified transitions of the State attribute of an Event Condition object. A client MMS-user receiving, as a result of an Event Enrollment for the specified transition, a notification of a state transition, may acknowledge receipt of the transition notification to the VMD. A client MMS-user acknowledging receipt of an event transition notification may further, through the use of an "Acknowledgement Event Condition," trigger a specified network-triggered Event Condition object, the result of which notifies other client MMS-users of the event of receipt of the acknowledgement by the VMD. An Event Enrollment shall be represented by the Event Enrollment object.

15.1.3.1 Attributes of Event Enrollment Objects

The attributes of the Event Enrollment object shall be defined as follows:

Object: Event Enrollment

```
Key Attribute: Event Enrollment Name
Attribute: MMS Deletable (TRUE, FALSE)
Attribute: Enrollment Class (MODIFIER, NOTIFICATION)
Attribute: Event Condition Reference
Attribute: Event Condition Transitions (DISABLED-TO-ACTIVE,
           DISABLED-TO-IDLE, IDLE-TO-ACTIVE, IDLE-TO-DISABLED,
           ACTIVE-TO-IDLE, ACTIVE-TO-DISABLED, ANY-TO-DELETED)
Attribute: Application Association Local Tag
Constraint: Enrollment Class = MODIFIER
    Attribute: Invoke ID
    Attribute: Remaining Acceptable Delay
Constraint: Enrollment Class = NOTIFICATION
    Attribute: Notification Lost (TRUE, FALSE)
    Attribute: Event Action Reference
    Attribute: Acknowledgement Event Condition Reference
    Attribute: Duration (CURRENT, PERMANENT)
    Attribute: Client Application
    Attribute: Additional Detail
    Attribute: Alarm Acknowledgement Rule (NONE, SIMPLE,
               ACK-ACTIVE, ACK-ALL)
    Attribute: Time Active Acknowledged
    Attribute: Time Idle Acknowledged
    Attribute: State (DISABLED, IDLE, ACTIVE, IDLE-NO-ACK-I,
               IDLE-NO-ACK-A, ACTIVE-NO-ACK-A, ACTIVE-ACKED,
               IDLE-ACKED)
```

Event Enrollment Name

The Event Enrollment Name attribute shall be the object name used to identify the Event Enrollment object.

MMS Deletable

The MMS Deletable attribute shall indicate whether (true) or not (false) the Event Enrollment object may be deleted using the DeleteEventEnrollment service.

Enrollment Class

The Enrollment Class attribute shall indicate the class of the Event Enrollment object. Two values for the Enrollment Class attribute are defined:

MODIFIER - The Event Enrollment object represents a temporary (executed one time only and deleted) Event Enrollment, created as a result of receipt of a service indication specifying a confirmed MMS service which was modified by the Attach To Event Condition service modifier. The effect of the Attach To Event Condition service modifier on the Transaction object is specified in this part of ISO/IEC 9506, 7.2.11. An Event Enrollment object having the value of the Enrollment Class attribute equal to MODIFIER is referred to as a modifier Event Enrollment object. Modifier Event Enrollment objects shall not be alterable through the AlterEventEnrollment service or deletable through the DeleteEventEnrollment service.

NOTIFICATION - The Event Enrollment object is an explicitly defined or predefined Event Enrollment object. It requests that the Procedure for Event Transition Processing (see 15.1.4.2) be executed upon detection of any of the indicated Event Condition transitions. An Event Enrollment object having the value of the Enrollment Class attribute equal to NOTIFICATION is referred to as a notification Event Enrollment object.

Event Condition Reference

This attribute identifies by reference the Event Condition object, which will, through transitions of the State attribute, cause invocation of the Procedure for Event Transition Processing (see 15.1.4.2) for this Event Enrollment object. If the referenced Event Condition object becomes unavailable for use, for example the referenced Event Condition object is deleted as a result of a Domain deletion or application association termination, the value of this attribute shall become UNDEFINED.

Event Condition Transitions

When the Event Enrollment object references a monitored Event Condition, the Event Condition Transitions attribute shall contain the set of Event Condition transitions which is to invoke the Procedure for Event Transition Processing (see 15.1.4.2) for this Event Enrollment object. It shall consist of any non-empty set composed of members chosen from the elements DISABLED-to-ACTIVE, DISABLED-to-IDLE, IDLE-to-ACTIVE, IDLE-to-DISABLED, ACTIVE-to-IDLE, ACTIVE-to-DISABLED, ANY-to-DELETED. For a notification Event Enrollment object, if the value of the Event Enrollment object's Alarm Acknowledgement Rule attribute is not equal to NONE, then this attribute shall include transitions to the ACTIVE state. If the ACK-ALL Alarm Acknowledgement Rule is specified, then this attribute shall also include transitions to the IDLE state.

When the Event Enrollment object references a network-triggered Event Condition object, this attribute shall specify the empty set and the Procedure for Event Transition Processing (see 15.1.4.2) for this Event Enrollment shall be executed only when the event is explicitly triggered by an MMS-client using the TriggerEvent service, when the network-triggered event occurs as a result of an autonomous VMD action, or when the specified network-triggered Event Condition object is utilized as an Acknowledgement Event Condition and is triggered by an MMS-client using the AcknowledgeEventNotification service.

Application Association Local Tag

The Application Association Local Tag attribute identifies locally the application association and abstract syntax which shall be used for:

a) EventNotifications in the case of notification Event Enrollment objects, and

b) providing the value of the Application Association Identifier attribute of the Transaction object created during execution of MMS services modified by the AttachToEventCondition modifier.

This attribute cannot be reported through any of the MMS services. The Abstract Syntax component of this attribute shall be:

a) the abstract syntax in use on the application association over which the DefineEventEnrollment service request or AttachToEventCondition modifier was received, or

b) as specified locally if the Event Enrollment object is created outside of MMS.

If the application association terminates, or never existed (i.e. an Event Enrollment object was created on behalf of a "third party," or an MMS-User other than the MMS-User creating this Event Enrollment object, and for which there was not a current application association), the value for the Application Association Local Identifier component of this attribute shall be UNDEFINED.

Invoke ID

The Invoke ID attribute shall exist only for modifier Event Enrollment objects. It shall contain the Invoke ID of the Transaction object (described in 7.2.11) of the modified service.

Remaining Acceptable Delay

The Remaining Acceptable Delay attribute shall exist only for modifier Event Enrollment objects. It shall contain either the value FOREVER or the remaining duration of time in seconds for which the MMS-user requesting the modifier Event Enrollment object is willing to wait for the specified Event Condition transitions to occur. If more than one modifier Event Enrollment object is specified for an Event Condition object, this attribute shall indicate separately the remaining acceptable delay for each modifier Event Enrollment object following commencement of modifier processing.

When a modifier Event Enrollment object is created and the Acceptable Delay parameter is provided, the value of the Acceptable Delay parameter is substituted into the value of the Remaining Acceptable Delay attribute. The VMD shall manage the value of this attribute as a representation of a timer, where the decrementation from the initial value begins immediately following the beginning of modifier processing under control of the Transaction object (see 7.2.10.) If the value of this attribute becomes zero, the Procedure for Event Enrollment Deletion (15.13.2.4) shall be followed and the influence of the modifier Event Enrollment object on the Transaction object shall no longer be considered by the VMD.

Notification Lost

The Notification Lost attribute shall exist only for notification Event Enrollment objects. The value of this attribute shall be true during periods of time when invocation of the Procedure for Event Transition Processing (see 15.1.4.2) has been modified due to resource limitations at the VMD, or if a notification cannot be sent due to the inability of the VMD to establish an application association for Event Enrollment objects having a value for the Duration attribute of PERMANENT. Otherwise, the value of the Notification Lost attribute shall be false. During the interval in which the Procedure for Event Transition Processing (see 15.1.4.2) has been modified due to lack of resources at the VMD, transitions of the Event Condition object's State attribute shall not invoke additional Transition Processing procedures.

Event Action Reference

The Event Action Reference attribute shall exist only for notification Event Enrollment objects. Its value may be UNSPECIFIED, indicating that the Event Enrollment requests only an EventNotification, or it may contain a reference to an Event Action object, indicating that the Event Enrollment requests an EventNotification containing the result of execution of the specified Event Action. In the case that the object referenced by the Event Action Reference attribute becomes unavailable, for example as a result of deletion of a Domain or termination of an application association, the value of this attribute shall become UNDEFINED.

Acknowledgement Event Condition Reference

The Acknowledgement Event Condition Reference attribute shall exist only for notification Event Enrollment objects. Its value may be UNSPECIFIED, indicating that an acknowledgement Event Condition has not been specified for the Event Enrollment object, or it may contain a reference to a network-triggered Event Condition object, which shall serve as the acknowledgement Event Condition. In the case that the object referenced by the Acknowledgement Event Condition Reference becomes unavailable, for example through the result of deletion of a domain or termination of an application association, the value of this attribute shall become UNDEFINED.

Duration

The Duration attribute shall exist only for notification Event Enrollment objects. Its value shall indicate the duration of the Event Enrollment object and may have either of two values.

CURRENT - shall indicate that the Event Enrollment object is defined for the life of the application association over which the Event Enrollment object was defined. A reference to this application association shall be contained in the Application Association Local Identifier component of the Application Association Local Tag attribute.

PERMANENT - shall indicate that the Event Enrollment object is defined for the life of the VMD unless explicitly deleted.

Client Application

The Client Application attribute shall only exist for notification Event Enrollment objects. It shall contain the identification of the enrolled client application and shall be of type Application Reference.

Additional Detail

The Additional Detail attribute shall exist for notification Event Enrollment objects only. The Additional Detail attribute shall maintain application and device-class specific state information. The semantics for this attribute shall be specified by a Companion Standard. If the abstract syntax defined in this part of ISO/IEC 9506 has been used for definition of the Event Enrollment object, then this attribute shall not exist.

Alarm Acknowledgement Rule

NOTE 1 – In the following discussions, "required" acknowledgements are acknowledgements which are required by ISO/IEC 9506. "Allowed" acknowledgements are acknowledgements which may, at the discretion of the MMS-user issuing the acknowledgement, be transmitted to the MMS-user issuing the Event Notification, but they should have the effect only of updating the Time Active Acknowleged attribute, or Time Idle Acknowledged attribute of the specified Event Enrollment object, or invoking the Procedure for Event Transition Processing (see 15.1.4.2) for an Acknowledgement Event Condition if one has been specified.

The Alarm Acknowledgement Rule attribute shall exist only for notification Event Enrollment objects which reference a monitored Event Condition object. It shall indicate the level of acknowledgement required for EventNotification service instances generated as a result of the Event Enrollment. The value of this attribute shall determine whether or not the Event Enrollment object shall be considered for inclusion in alarm enrollment summaries. This attribute shall contain one of the following values:

NONE - The Event Enrollment object shall not be included in alarm enrollment summaries. Acknowledgements to EventNotifications under the auspices of these Event Enrollment objects are allowed, but have no effect on the state of the Event Enrollment object.

NOTE 2 – The event transition should not represent a situation which requires acknowledgement from the enrolled client.

SIMPLE - Acknowledgement shall be allowed but not required for notifications specifying a transition to the ACTIVE state, and shall have an effect on the State attribute of the Event Enrollment object. Acknowledgement for notifications specifying a transition to the IDLE state shall be allowed and shall have no effect on the State attribute of the Event Enrollment object.

The attributes of this Event Enrollment object shall be included in alarm enrollment summaries unless the filter criteria eliminate it.

ACK-ACTIVE - Acknowledgement shall be required for notifications specifying a transition to the ACTIVE state. Acknowledgement shall be allowed for notifications specifying a transition to the IDLE state but shall have no effect on the State attribute of the Event Enrollment object. This Event Enrollment shall be included in alarm enrollment summaries unless the filter criteria eliminate it.

ACK-ALL - Acknowledgement shall be required for all notifications specifying a transition to the ACTIVE or the IDLE state. This Event Enrollment shall be included in alarm enrollment summaries unless the filter criteria eliminate it.

The value of the State attribute and the value of the Alarm Acknowledgement Rule attribute shall determine the complexity of the state table of the Event Enrollment object (see 15.22).

NOTE 3 – Acknowledgement should never be allowed for an event notification specifying the DISABLED state.

Time Active Acknowledged

The Time Active Acknowledged attribute shall only exist for notification Event Enrollment objects which reference a monitored Event Condition object. The value of the Time Active Acknowledged attribute shall only be meaningful when the the value of the Event Enrollment object's Alarm Acknowledgement Rule is not equal to NONE. The Time Active Acknowledged attribute shall record the time (date and time of day or Time Sequence Identifier) at which an acknowledgement for the most recently detected transition of the Event Condition object to the ACTIVE state was received from the enrolled client. If this acknowledgement has not been received, the value of the Time Active Acknowledged attribute shall be UNDEFINED.

Time Idle Acknowledged

The Time Idle Acknowledged attribute shall exist only for notification Event Enrollment objects which reference a monitored Event Condition object. The value of the Time Idle Acknowledged attribute shall only be meaningful when the value of the Event Enrollment object's Alarm Acknowledgement Rule is not equal to NONE. The Time Idle Acknowledged attribute shall record the time (date and time of day or Time Sequence Identifier) at which an acknowledgement for the most recently detected transition of the Event Condition object to the IDLE state was received. If this acknowledgement has not been received, the value of the Time Idle Acknowledged attribute shall be UNDEFINED.

State

The State attribute shall maintain the current state of the Event Enrollment object. At any instant an Event Enrollment object has a major state, which is equal to the state of the Event Condition object referenced by the Event Condition Reference attribute of the Event Enrollment object. If the referenced Event Condition object has become unavailable, either through the deletion of a domain or the loss of an application association, the major state shall be equal to the last known state of the Event Condition object.

Depending upon the value of the Event Enrollment object's Alarm Acknowledgement Rule attribute, the Event Enrollment object may also have an implied minor state. The minor state shall indicate whether or not the Event Enrollment object is waiting for a required acknowledgement from the client for an Event Condition object transition to the ACTIVE or the IDLE state. The meaning of the various minor states is as follows: NO-ACK-A - The minor state NO-ACK-A shall indicate that a required acknowledgement for a transition of the Event Condition object to the ACTIVE state has not been received from the client. This minor state is defined for the ACTIVE major state of Event Enrollment objects which specify an Alarm Acknowledgement Rule attribute value equal to SIMPLE, ACK-ACTIVE or ACK-ALL. It is also defined for the IDLE major state of Event Enrollment objects which specify a value for the Alarm Acknowledgement Rule attribute equal to ACK-ACTIVE or ACK-ALL.

When in the ACTIVE major state, the NO-ACK-A minor state shall indicate that the value of the Event Enrollment object's Time Active Acknowledged attribute is either UNDEFINED or is earlier than the value of the Event Condition object's Time Of Last Transition To Active attribute.

When in the IDLE major state, the NO-ACK-A minor state shall indicate that the value of the Event Condition object's Time Of Last Transition To Active attribute is not equal to UNDEFINED and that the value of the Event Enrollment object's Time Active Acknowledged attribute is either UNDEFINED or earlier in time than the value of the Event Condition object's Time Of Last Transition To Active attribute.

NO-ACK-I - The minor state NO-ACK-I shall indicate that a required acknowledgement for a transition of an Event Condition object to the IDLE state has not been received from the client. This minor state is only defined for the IDLE major state of Event Enrollment objects which contain a value for the Alarm Acknowledgement Rule attribute equal to ACK-ALL. It shall indicate that the required acknowledgement for a previous transition to the ACTIVE state, if any, has been received and that the value of the Event Enrollment object's Time Idle Acknowledged attribute is either UNDEFINED or is earlier than the value of the Event Condition object's Time Of Last Transition To Idle attribute.

ACKED - The minor state ACKED shall indicate that all required acknowledgements have been received from the client.

The structure of the State attribute shall consist of three components: the value of the major state, the value of the minor state, and a Reportable component, which is reported to the MMS Client through the MMS services.

The value of the Reportable component of the State attribute shall be equal to the major state if no minor state is implied, otherwise the value of Reportable component of the State attribute shall be equal to the concatenation of the major and minor states.

15.1.3.2 Operations on Event Enrollment

The services and modifier which operate upon Event Enrollment objects are:

a) DefineEventEnrollment - This service may be used to create a notification Event Enrollment Object.

b) AlterEventEnrollment - This service may be used to modify certain attributes of a notification Event Enrollment object referencing a monitored Event Condition object.

c) DeleteEventEnrollment - This service may be used to delete one or more notification Event Enrollment objects.

d) GetEventEnrollmentAttributes - This service obtains the descriptive attributes of one or more Event Enrollment objects.

e) ReportEventEnrollmentStatus - This service obtains the current status attributes of an Event Enrollment object.

f) EventNotification - This service notifies a client of an event condition transition for which the client is currently or permanently enrolled to receive notifications.

g) AcknowledgeEventNotification - This service updates an Event Enrollment object's Time Active Acknowledged attribute, or Time Idle Acknowledged attribute, or both, unless the value of the Alarm Acknowledgement Rule attribute is equal to NONE.

h) GetAlarmSummary - This service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current status of monitored Event Condition objects and related attributes of their referenced notification Event Enrollment objects.

i) GetAlarmEnrollmentSummary - This service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current alarm status of notification Event Enrollment objects and related attributes of their referenced monitored Event Condition objects.

j) Attach To Event Condition - This service modifier creates a modifier Event Enrollment.

15.1.4 Event Detection and Notification

A VMD which supports the event management model shall be responsible for servicing events which are defined at the VMD. This requires that the VMD detect state changes for enabled monitored Event Condition objects, detect the deletion of referenced Event Condition objects, detect the occurrence of autonomous network-triggered events or accept requests to trigger network-triggered Event Condition objects, or any combination of the above actions and process these changes (or trigger requests) in order to execute the Procedure for Event Transition Processing (see 15.1.4.2) for each client which has enrolled for the state change (or for the TriggerEvent request.)

The procedures which specify the requirements for event monitoring and transition processing are given below.

15.1.4.1 Procedure for Event Condition Monitoring

The VMD shall be responsible for detecting changes (transitions) in the state of enabled monitored Event Condition objects (whether defined by DefineEventCondition, or by local means), detecting occurrence of autonomous network-triggered events, and for accepting requests to trigger network-triggered Event Condition objects. When any of these occur, the procedure for event transition processing (15.1.4.2) shall be executed.

Determination of the value of the State attribute of an enabled monitored Event Condition object requires evaluation of the Event Condition object's state on a timely basis following a change in the determined value of the referenced boolean variable, where the value of the Monitored Variable Reference attribute is not equal to UNSPECIFIED. In this case the monitored variable is represented by the Variable object referenced in the Monitored Variable Reference attribute of the Event Condition object. In the case where the value of the Monitored Variable Reference attribute is equal to UNSPECIFIED, the determination of the value of the State attribute shall be a local matter. The decision as to when state evaluation is to be accomplished shall be a local (to the VMD) matter. The Priority attribute and the Evaluation Interval attribute of the Event Condition object are provided as guidance in making this decision.

15.1.4.2 Procedure for Event Transition Processing

The procedure for Event Transition processing shall be executed:

1) when a change in the state (to ACTIVE or IDLE) of an enabled monitored Event Condition object is detected;

2) when the value of the Enabled attribute of a monitored Event Condition object is changed from true to false (by use of the AlterEventConditionMonitoring service, or by local means);

3) when a request to trigger a network-triggered Event Condition object is received;

4) when an autonomous network-triggered event occurs;

5) when the Event Condition Transitions attribute of an Event Enrollment object contains the value ANY-TO-DELETED and the value of the Event Condition Reference attribute becomes UNDEFINED (as a result of deletion of the referenced Event Condition object); or,

6) when the Event Condition Transitions attribute of an Event Enrollment object contains the value ANY-TO-DELETED and the value of the Monitored Variable Reference attribute of a monitored Event Condition object, referenced by the Event Condition Reference attribute of the Event Enrollment object, becomes UNDEFINED (as a result of deletion of a domain or loss of an application association).

This procedure shall involve the following actions:

NOTE 1 – The mechanisms described in this clause describe the logical operation required for the Procedure for Event Transition Processing. Implementations may choose other methods of internal operation, as long as the externally visible characteristics described in this clause are maintained.

a) The Procedure for Event Condition Object Update (15.1.4.2.1) shall be executed.

b) The Procedure for Event Condition Object Attribute Value Capture (15.1.4.2.2) shall be executed.

c) The Procedure for Event Enrollment Object Attribute Value Capture (15.1.4.2.3) shall be executed.

d) The Procedure for Event Enrollment Object Update (15.1.4.2.4) shall be executed.

e) The Procedure for Event Action Execution (15.1.4.2.5) shall be executed.

f) The Procedure for Establishment of an Application Association for Notification (15.1.4.2.6) shall be executed.

g) The Procedure for Invoking an Event Notification (15.1.4.2.7) shall be executed.

NOTE 2 – The actual invocation of the EventNotification service (including the execution of the applicable Event Action, if any) should occur as the captured information is processed, on a timely basis, in the sequence (within a given priority) in which it was captured.

Each of the above procedures shall be executed as an atomic entity, to the extent that a specific procedure or, where a procedure includes multiple iterations, each iteration shall succeed or fail as a discrete element. Failure of a specific procedure or iteration is defined as the inability to successfully complete the procedure, for any reason.

NOTE 3 – The action to take when there are multiple transitions from different Event Condition objects to process concurrently, or when an additional transition of the Event Condition object currently being processed is detected during and prior to completion of the Procedure for Event Transition Processing, is a local matter. The Priority and Severity attributes are provided as guidance to the VMD in this matter, and the Notification Lost attribute is provided as a means for the VMD to declare to the client its inability to completely process the detected transitions.

It is possible that due to resource limitations at the VMD, it may be temporarily not possible to complete a specific procedure or iteration. In this case, the VMD may declare the procedure or iteration to have failed, or may suspend processing of the procedure or iteration until resources become available. The determination of whether to declare failure or temporarily suspend processing shall be a local matter. The decision criteria used in making this determination shall be reflected in the PICS, as defined in ISO/IEC 9506-2, clause 18.

For a given Event Enrollment, if the Procedure for Event Action Execution is completed successfully, then the Procedure for Invoking an Event Notification shall be executed. Logically, the effect of failure of any specific procedure is specified below.

a) Failure of the Procedure for Event Condition Object Update shall result in complete failure of the Procedure for Event Transition Processing. The value of the Notification Lost attribute for all Event Enrollment objects referenced by the List Of Event Enrollment Reference attribute of the Event Condition object shall, if possible, be set to true.

b) Failure of the Procedure for Event Condition Object Attribute Value Capture shall result in complete failure of the Procedure for Event Transition Processing. The value of the Notification Lost attribute for all Event Enrollment objects referenced by the List Of Event Enrollment Reference attribute of the Event Condition object shall, if possible, be set to true.

c) Failure of any iteration of the Procedure for Event Enrollment Object Attribute Value Capture shall result in cancellation of the EventNotification service for the enrolled client. The value of the Notification Lost attribute of the specific Event Enrollment object shall, if possible, be set to true. If the List Of Event Enrollment Reference attribute of the Event Condition object contains only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall be considered to have failed.

d) Failure of any iteration of the Procedure for Event Enrollment object Update shall result in cancellation of the EventNotification service for the enrolled client. The value of the Notification Lost attribute of the specific Event Enrollment object shall, if possible, be set to true. If the List Of Event Enrollment Reference attribute of the Event Condition object should contain only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall be considered to have failed.

e) Failure of an iteration of the Procedure for Event Action Execution shall result in the value false appearing in the Success or Failure parameter of the EventNotification service, when issued. Failure of the Procedure for Event Action Processing shall not otherwise affect the Procedure for Event Transition Processing.

f) Failure of an iteration of the Procedure for Establishment of an Application Association for Notification shall result in cancellation of the EventNotification service for the enrolled client. The value of the Notification Lost attribute of the specific Event Enrollment object shall, if possible, be set to false. If the List Of Event Enrollment Reference attribute of the Event Condition object should contain only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall be considered to have failed.

g) Failure of an iteration of the Procedure for Invoking an Event Notification shall result in cancellation of the EventNotification service for the enrolled client. The value of the Notification Lost attribute of the specific Event Enrollment object shall, if possible, be set to false. If the List Of Event Enrollment Reference attribute of the Event Condition object should contain only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall be considered to have failed.

15.1.4.2.1 Procedure for Event Condition Object Update

This procedure shall be executed following determination of a change in the value of the State attribute of the Event Condition object. The procedure involves the following actions:

a) Capture the time of determination (time of day or Time Sequence Identifier) of the new value of the State attribute.

b) Alter the Event Condition object's State attribute to the new state, as required.

c) If the Event Condition object is a monitored Event Condition object and the new state is ACTIVE, replace the Time Of Last Transition To Active attribute value with the captured time (date and time of day or Time Sequence Identifier).

d) If the Event Condition object is a monitored Event Condition object and the new state is IDLE, replace the Time Of Last Transition To Idle attribute with the captured time (date and time of day or Time Sequence Identifier).

15.1.4.2.2 Procedure for Event Condition Object Attribute Value Capture

This procedure shall be executed to capture the values of the attributes of the Event Condition object which are required in order to issue the EventNotification service. The values of the following attributes of the Event Condition object shall be captured:

a) Event Condition Name.

b) The value of the State attribute.

c) The value of the Severity attribute.

15.1.4.2.3 Procedure for Event Enrollment Object Attribute Value Capture

This procedure shall be executed for each enrolled Event Enrollment object which specifies the current transition of the Event Condition object in the value of the Event Condition Transitions attribute. For each Event Enrollment object specifying the value of the current transition in the Event Condition Transitions attribute, the values of the following attributes of the Event Enrollment object shall be captured:

a) Event Enrollment Name.

b) Application Association Local Tag

c) Notification Lost

d) Event Action Reference

e) Client Application

f) Alarm Acknowledgement Rule

The time of transition captured during the Procedure for Event Condition Object Update and the attribute values captured from the Event Condition and Event Enrollment objects shall be retained until the Procedure for Invoking the Event Notification Service (15.1.4.2.7) is completed.

15.1.4.2.4 Procedure for Event Enrollment Object Update

For each Event Enrollment object referenced by the Event Condition object's List Of Event Enrollment Reference attribute and specifying the current transition, one of the following sub-procedures shall be executed.

a) If the Event Enrollment object is a notification Event Enrollment object, then:

   1) if the Event Condition object's new State attribute value is ACTIVE, replace the Event Enrollment object's Time Active Acknowledged attribute with the value UNDEFINED;

   2) if the Event Condition object's new State attribute value is IDLE, replace the Event Enrollment object's Time Idle Acknowledged attribute with the value UNDEFINED;

b) if the Event Enrollment object is a modifier Event Enrollment object, continue processing of the Transaction object as defined in 7.2.10.2.

   NOTE – Transition processing should not wait for processing of the Transaction object.

15.1.4.2.5 Procedure for Event Action Execution

When the Event Action Reference attribute of an Event Enrollment object contains a reference to an Event Action object, the VMD shall create a Transaction object (see 7.2.10) containing the following elements:

a) a unique locally assigned Invoke ID attribute,

b) a locally assigned Application Association Identifier,

c) a List Of Pre-Execution Modifier attribute equal to the Event Action object's List Of Modifier attribute,

d) a Current Modifier Reference attribute initialized to refer to the first modifier in the List of Pre-Execution Modifier attribute,

e) a Confirmed Service Request attribute equal to the value of the Event Action object's Confirmed Service Request attribute,

f) an empty List Of Post-execution Modifier attribute and a Cancelable attribute initialized with a value equal to false.

The procedure for transaction processing (see 7.2.10.2) shall then be executed. The result of the execution of this procedure shall be used to supply the value of the Success Or Failure parameter in the Event Notification request service primitive and either the Confirmed Service Response parameter or the Confirmed Service Error parameter (depending on success or failure of the confirmed service) of the Action Result parameter of the EventNotification service request.

NOTE – Transaction processing results in the consumption of resources and can potentially result in error conditions related to insufficient resources if care is not taken during application design. In addition, the Transaction object created, as a result of an Event Action execution, reduces the number of available outstanding confirmed service invocations, as limited by the maximum number of services outstanding parameters negotiated through the Initiate service as described in 8.2.1.2. If this limit is exceeded, the result of the Event Action (upon attempt to create the Transaction object) may be a service error specifying a resource problem.

If the value of the Event Action Reference attribute of the Event Enrollment object has become UNDEFINED as a result of Domain deletion or loss of an application association, the value of the Success Or Failure parameter of the EventNotification service request shall be false and the value of the Confirmed Service Error parameter of the EventNotification service request shall be OBJECT-UNDEFINED.

15.1.4.2.6 Procedure for Establishment of an Application Association for Notification

For each Event Enrollment object specifying the value of the current transition in the Event Condition Transitions attribute, the application association to be used for transmission of EventNotification service primitives issued as a result of a notification Event Enrollment shall be established as specified below.

a) If the value of the Duration attribute of the Event Enrollment object is equal to CURRENT, then the application association and abstract syntax specified by the Application Association Local Tag attribute of the Event Enrollment object shall be used for invoking the EventNotification service. If this application association terminates for any reason, the Event Enrollment object, and all pending event notifications for it, shall be deleted.

b) If the Duration attribute of the Event Enrollment object is equal to PERMANENT, then the application association to be used shall be determined as follows:

1) The application association identified by the Application Association Local Tag attribute of the Event Enrollment object shall be used if:

i) the application association still exists;

ii) the application association is with the client specified by the Client Application attribute of the Event Enrollment object and uses the abstract syntax specified in the Application Association Local Tag attribute;

iii) the client supports VMD initiation of the EventNotification service.

2) Otherwise, any currently active application association between the VMD and the enrolled client using the abstract syntax specified in the Application Association Local Tag attribute of the Event Enrollment object shall be used, as long as initiation of the EventNotification service by the VMD has been negotiated for the application association.

3) If no suitable application association exists between the VMD and the enrolled client, then the VMD shall attempt to establish a suitable application association with the enrolled client using the abstract syntax specified in the Application Association Local Tag attribute of the Event Enrollment object (see the Initiate service). If this succeeds, the newly established application association shall be used.

If it is not possible to establish an application association for the Event Enrollment, then the VMD may attempt to retry the establishment process on a periodic basis, or it may delete the Event Enrollment object and all pending event notifications requested by it. The action to be taken shall be a local decision of the VMD.

NOTE – Deletion of the Event Enrollment object should be performed as a last resort.

15.1.4.2.7 The Procedure for Invoking an Event Notification

For each Event Enrollment object specifying the value of the current transition in the Event Condition Transitions attribute, this procedure shall be executed in order to process the captured information and issue an EventNotification service request.

After determining the result of the execution of the Event Action (if applicable, see 15.1.4.2.5,) and using the application association with the client (see 15.1.4.2.6), the EventNotification service request primitive shall be issued. The derivation of the parameters of the EventNotification service request is specified below.

a) The Event Enrollment Name parameter shall be equal to the value captured from the Event Enrollment Name attribute of the Event Enrollment object.

b) The Event Condition Name parameter shall be equal to the value captured from the Event Condition Name attribute of the Event Condition object.

c) The Severity parameter shall be equal to the value captured from the Severity attribute of the Event Condition object.

d) The Current State parameter shall be equal to the value captured from the State attribute of the Event Condition object.

e) The Transition Time parameter shall be equal to the time captured when the transition of the Event Condition object's State attribute was determined.

f) The Notification Lost parameter shall be equal to the value captured from the Notification Lost attribute of the Event Enrollment object.

g) The Alarm Acknowledgement Rule parameter shall be equal to the value captured from the Alarm Acknowledgement Rule attribute of the Event Enrollment object.

h) The Action Result parameter, and the included sub-parameters of Event Action Name, Success Or Failure, Confirmed Service Response or Confirmed Service Error, shall be derived from the result of executing the Event Action specified by the referenced Event Action object, if any. If no Event Action object is referenced by the Event Enrollment object, these parameters shall not be included in the EventNotification service request.

Following successful completion of an iteration of the Procedure for Invoking an Event Notification, the value of the Notification Lost attribute of the specific Event Enrollment object shall be set to false.

NOTE – There is an implied time skew between the time in the notification and the time of the values reported by the Event Action result. Implementations should attempt to minimize this time.

15.1.4.3 Procedure For Acknowledgement of Event Notifications

Acknowledgements for Event Notifications initiated during the Procedure for Event Transition Processing shall take one of two forms:

A simple acknowledgement shall consist of an acknowledgement for a specified transition, within the scope of a single Event Enrollment object.

A group acknowledgement shall provide the capability to allow an acknowledgement from any member of a specified group of Event Enrollment objects to suffice for required acknowledgements from all members of the group, and shall further provide a notification to all members of the group that acknowledgements have been received.

Group enrollments which support group acknowledgements shall be represented as multiple Event Enrollment objects which reference a transition of a monitored or network-triggered Event Condition object associated with the application to which the VMD has been applied. This Event Condition object shall be referred to as the "process" Event Condition object. Group Enrollments shall additionally reference a network-triggered Event Condition object which shall serve as an "acknowledgement" Event Condition object, where the receipt of an acknowledgement from any member of the group is treated as a distinct event. An MMS-User acknowledging an Event Notification conveying the transition of a process Event Condition object may optionally specify the name attribute of an acknowledgement Event Condition object which, when triggered, results in the execution of the Procedure for Event Transition Processing, (see 15.1.4.2) for enrollees of the acknowledgement Event Condition object, communicating via Event Notifications the fact that one member of the enrolled group has acknowledged the transition of the process Event Condition object. Figure 14 shows the relationships between the process Event Condition object, Event Enrollment objects, and the acknowledgement Event Condition object.

When an acknowledgement is received (via the AcknowledgeEventNotification service) for a transition of an Event Condition object, The VMD shall consider the Event Enrollment object for which the acknowledgement was received to be an "acknowledged" Event Enrollment object, and shall execute the following procedure for the acknowledged Event Enrollment object.

a) If the attributes of the acknowledged Event Enrollment object indicate that an acknowledgement for the specified state change (both state and time) has already been received, no action shall be taken.

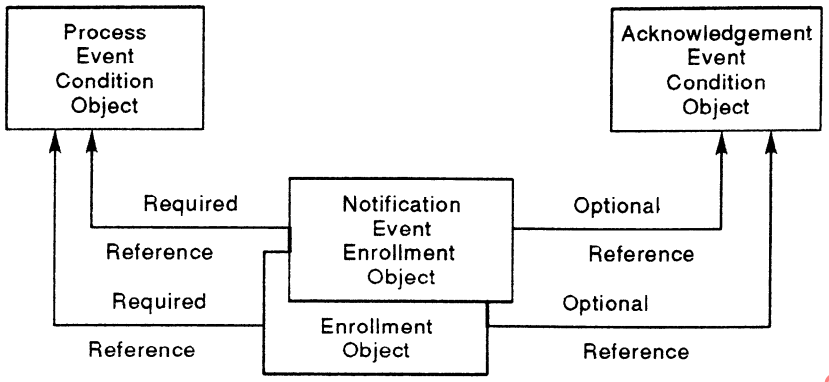

Process Event Condition Object
Acknowledgement Event Condition Object
Required
Reference
Required
Reference
Notification Event Enrollment Object
Enrollment Object
Optional
Reference
Optional
Reference

Figure 14 – Relationships Between Process Event Condition Objects, Notification Event Enrollment Objects and Acknowledgement Event Condition Objects

NOTE – This may occur due to the possibility of multiple invocations of the EventNotification being issued by the VMD.

b) Otherwise, if the Acknowledged State parameter is not equal to the current value of the State attribute of the Event Condition object referenced by the acknowledged Event Enrollment object, or if the Time Of Acknowledged Transition parameter is not equal to the current value of the Time Of Last Transition To Active attribute or the Time Of Last Transition To Idle attribute (as applicable to the Acknowledged State) of the Event Condition object referenced by the acknowledged Event Enrollment object, then no action shall be taken.

c) Otherwise, if the value of the Acknowledged State parameter is ACTIVE, then the acknowledged Event Enrollment object's Time Active Acknowledged attribute shall be replaced by the current time (date and time of day or Time Sequence Identifier).

d) Otherwise, if the value of the Acknowledged State parameter is IDLE, then the acknowledged Event Enrollment object's Time Idle Acknowledged attribute shall be replaced by the current time (date and time of day or Time Sequence Identifier).

e) If the value of the Alarm Acknowledgement Rule attribute of the acknowledged Event Enrollment object is not equal to SIMPLE or NONE, and the acknowledged transition corresponds to a required acknowledgement, the requirement shall be considered satisfied.

f) If the Acknowledgement Event Condition parameter has been provided, the network-triggered Event Condition object specified by the value of this parameter shall be located and triggered. The Procedure for Event Transition Processing, (see 15.1.4.2) shall be followed for the processing of the network-triggered Event Condition object thus triggered.

g) If the Acknowledgement Event Condition parameter has been provided, the VMD shall search the List Of Event Enrollment Reference attribute of the specified network-triggered Event Condition object, and (with the exception of the acknowledged Event Enrollment object) for each object referenced by the list the VMD shall:

1) Replace the value of the Time Active Acknowledged attributed, or the value of the Time Idle Acknowledged attribute, as appropriate, using the value and attribute name specified in the foregoing procedure for the acknowledged Event Enrollment object. If the foregoing procedure resulted in no change to either the Time Active Acknowledged attribute or the Time Idle Acknowledged attribute, then no action is taken.

2) Eliminate any requirements for acknowledgement for notification of the specified transition of the process Event Condition object which may exist.

15.2 DefineEventCondition Service

The DefineEventCondition service shall provide a mechanism whereby a client MMS-user may request the creation of an Event Condition object at a VMD.

15.2.1 Structure

The structure of the component service primitives is shown in Table 68.

Table 68 – DefineEventCondition Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Condition Name		M	M(=)			
Class		M	M(=)			
Priority		M	M(=)			
Severity		M	M(=)			
Alarm Summary Reports		C	C(=)			
Monitored Variable		C	C(=)			
Evaluation Interval		C	C(=)			
Result(+)	(COMP)			S	S(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

15.2.1.1 Argument

This parameter shall convey the parameters of the DefineEventCondition service request.

15.2.1.1.1 Event Condition Name

The Event Condition Name parameter, of type Object Name, shall contain the name to be assigned to the created Event Condition object. It shall uniquely identify the Event Condition object.

15.2.1.1.2 Class

The Class parameter, of type EC Class, shall contain the value of the Event Condition object's Class attribute. The semantics and allowed values for this attribute are defined in 15.1.1.1.

15.2.1.1.3 Priority

The Priority parameter, of type Priority, shall contain the initial value of the Event Condition object's Priority attribute. The semantics and range of values for this attribute are defined in 15.1.1.1.

15.2.1.1.4 Severity

The Severity parameter, of type integer, shall contain the value of the Event Condition object's Severity attribute. The semantics and range of values for this attribute are described in 15.1.1.1.

15.2.1.1.5 Alarm Summary Reports

This parameter, of type boolean, shall be omitted for a network-triggered Event Condition object. It shall be specified for a monitored Event Condition object. When specified, the Alarm Summary Reports parameter shall contain the value of the Event Condition object's Alarm Summary Reports attribute. The semantics and allowed values for this attribute are defined in 15.1.1.1.

15.2.1.1.6 Monitored Variable

The Monitored Variable parameter, of type Variable Specification, shall be a required parameter for definition of a monitored Event Condition object. It shall be omitted for definition of a network-triggered Event Condition object. If specified, it shall specify a Named or Unnamed Variable object of abstract boolean type which shall be referenced by the Event Condition object's Monitored Variable Reference attribute. The semantics of this attribute are given in 15.1.1.1. The attributes of the Variable Specification type can be found in clause 12.5.2.

15.2.1.1.7 Evaluation Interval

The Evaluation Interval parameter, of type integer, shall be a required parameter for definition of a monitored Event Condition object, otherwise it shall be omitted. It shall specify the maximum acceptable time, in milliseconds, between determinations of the value of the Event Condition object's State attribute. The value of this parameter shall be such that determination at a periodic interval less than or equal to the specified interval shall be sufficient for reliable and timely detection of state changes. Any integral value between zero and 2**31 - 1, inclusive, may be specified.

This parameter shall be provided as guidance to the VMD. Acceptance of this value guarantees only that the VMD shall make every effort to honour the specified value. A VMD which determines itself, through local means, to be unable to honour the service request due to resource or other limitations shall return a TIME-RESOLUTION error in response to the service request.

The actual decision to determine the value of the Event Condition object's State attribute may be based on the passage of time (such as a scan cycle), on knowledge about the variable referenced by the Monitored Variable parameter, or other local criteria. If the decision is to be time-based, then a non-zero value for the Evaluation Interval parameter shall specify the maximum acceptable time between determinations. A zero value shall indicate that any interval shall be acceptable.

15.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall not supply service specific parameters.

15.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.2.2 Service Procedure

If the requested definition is acceptable, a new Event Condition object shall be created and initialized as described below. This object shall then become part of the state of the application association over which the requested definition was received, of the VMD, or of a Domain of the VMD, depending on the scope of the Event Condition Name parameter. Finally, a positive response shall be issued, indicating that the Event Condition object has been created.

The initial value for the attributes of the Event Condition object (as described in 15.1.1.1) are specified below:

a) Event Condition Name - Initialized to equal the value of the Event Condition Name parameter.

b) MMS Deletable - Initialized to the value true.

c) Event Condition Class - Initialized to equal the value of the Class parameter.

d) State - Initialized to the value DISABLED.

e) Priority - Initialized to equal the value of the Priority parameter.

f) Severity - Initialized to equal the value of the Severity parameter.

g) Additional Detail - This attribute shall only be specified for communication in the abstract syntax of a Companion Standard. If present, it shall be initialized in accordance with the requirements of the applicable Companion Standard when the DefineEventCondition service indication was received.

h) List of Event Enrollment Reference - Initialized to empty.

i) Enabled - For a monitored Event Condition object this attribute shall be initialized to the value false. This attribute shall not exist for a network-triggered Event Condition object.

j) Alarm Summary Reports - For a monitored Event Condition object this attribute shall be initialized to equal the value of the Alarm Summary Reports parameter. This attribute shall not exist for a network-triggered Event Condition object.

k) Monitored Variable Reference - For a monitored Event Condition object this attribute shall be initialized to reference the object specified by the value of the Monitored Variable parameter. This attribute shall not exist for a network-triggered Event Condition object.

l) Evaluation Interval - For a monitored Event Condition object this attribute shall be initialized to equal the value of the Evaluation Interval parameter. This attribute shall not exist for a network-triggered Event Condition object.

m) Time Of Last Transition To Active - For a monitored Event Condition object this attribute shall be initialized to the value UNDEFINED. This attribute shall not exist for a network-triggered Event Condition object.

n) Time Of Last Transition To Idle - For a monitored Event Condition object this attribute shall be initialized to the value UNDEFINED. This attribute shall not exist for a network-triggered Event Condition object.

15.3 DeleteEventCondition Service

The purpose for the DeleteEventCondition service is to allow a client MMS-user to request that a VMD delete one or more MMS defined Event Condition objects.

15.3.1 Structure

The structure of the component service primitives is shown in Table 69.

Table 69 – DeleteEventCondition Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Scope Of Delete		M	M(=)			
Event Condition Names		C	C(=)			
Domain Name		C	C(=)			
Result(+)	(COMP)			S	S(=)	
Candidates Not Deleted				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.3.1.1 Argument

This parameter shall convey the parameters of the DeleteEventCondition service request.

15.3.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete to be attempted. This parameter shall contain one of the following values:

SPECIFIC - Shall designate the MMS-defined Event Condition objects identified by the value of the Event Condition Names parameter as the only candidates for deletion.

AA-SPECIFIC - Shall designate all MMS defined Event Condition objects whose scope is the current application association as candidates for deletion.

DOMAIN - Shall designate all MMS defined Event Condition objects whose scope is the specified Domain as candidates for deletion.

VMD - Shall designate all MMS defined Event Condition objects whose scope is VMD as candidates for deletion.

Only those candidate Event Condition objects having an empty List Of Event Enrollment Reference attribute shall be deleted.

15.3.1.1.2 Event Condition Names

This parameter, containing a list of one or more Event Condition Name attributes, each of type Object Name and each identifying an Event Condition object, shall be specified if the value of the Scope Of Delete parameter is equal to SPECIFIC. Otherwise, it shall be omitted. It shall give the names of the specific candidate Event Condition objects to be deleted.

15.3.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if the value of the Scope Of Delete parameter is equal to DOMAIN. Otherwise, it shall be omitted. It shall give the name of the Domain for which all MMS defined Event Condition objects are candidates for deletion.

15.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

15.3.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain a count of the number of Event Condition objects which were included in the scope of Event Condition objects to be deleted, but which were not deleted due to a non-empty value of the List Of Event Enrollment Reference attribute, or a value of false in the MMS Deletable attribute.

15.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.3.2 Service Procedure

If the request is acceptable, for each Event Condition object having an Event Condition Name attribute of the specified scope, the VMD shall verify that the Event Condition object's List Of Event Enrollment Reference attribute is empty. If not empty, or if the MMS Deletable attribute has the value false, the VMD shall count the Event Condition object as a Candidate Not Deleted, and shall not delete the Event Condition object. Otherwise, the Event Condition object shall be deleted and its name shall be made available for immediate redefinition. If the deletion occurs as a result of deletion of a Domain or loss of an application association, and the List Of Event Enrollment Reference attribute contains a reference to an Event Enrollment object which specifies the transition ANY-TO-DELETED, the VMD shall execute the Procedure for Event Transition Processing (see 15.1.4.2) prior to deleting the Event Condition Object.

Finally, a positive response shall be issued, indicating the count of candidate Event Condition objects which were not deleted due to a non-empty List Of Event Enrollment Reference attribute, or a value of FALSE in the MMS Deletable attribute.

NOTE – If this count is not equal to zero, the requesting MMS-user (the client) may use the GetNameList service to determine the Event Condition objects which were not deleted.

Event Condition objects can also become deleted as a result of the deletion of a Domain or loss of an application association. In these cases, the Event Condition Reference attribute of any Event Enrollment object which referred to a deleted Event Condition object shall be set to the value UNDEFINED.

15.4 GetEventConditionAttributes Service

The GetEventConditionAttributes service shall provide a mechanism whereby a client MMS-user may obtain the attributes of an Event Condition object at a VMD.

15.4.1 Structure

The structure of the component service primitives is shown in Table 70.

Table 70 – GetEventConditionAttributes Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Condition Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
MMS Deletable				M	M(=)	
Class				M	M(=)	
Priority				M	M(=)	
Severity				M	M(=)	
Alarm Summary Reports				C	C(=)	
Monitored Variable				C	C(=)	
Evaluation Interval				C	C(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.4.1.1 Argument

This parameter shall convey the parameter of the GetEventConditionAttributes service request.

15.4.1.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the name of the Event Condition object for which the attributes are to be obtained.

15.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When a positive response is issued the following parameters are supplied.

15.4.1.2.1 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Event Condition object may be deleted using the DeleteEventCondition service.

15.4.1.2.2 Class

This parameter, of type EC Class, shall contain the value of the Event Condition object's Event Condition Class Attribute.

15.4.1.2.3 Priority

This parameter, of type Priority, shall contain the current value of the Event Condition object's Priority attribute.

15.4.1.2.4 Severity

This parameter, of type integer, shall contain the value of the Event Condition object's Severity attribute.

15.4.1.2.5 Alarm Summary Reports

For a monitored Event Condition object, this parameter, of type boolean, shall contain the value of the Event Condition object's Alarm Summary Reports attribute. This parameter shall be omitted for a network-triggered Event Condition object.

15.4.1.2.6 Monitored Variable

For a monitored Event Condition object, the Monitored Variable parameter, of type Variable Specification, shall contain the value of the key attribute from the object referenced by the Event Condition object's Monitored Variable Reference attribute, unless the value of the Monitored Variable Reference attribute is UNSPECIFIED, in which case this parameter shall be omitted. If the object referenced by the Monitored Variable Reference attribute becomes unavailable the Monitored Variable parameter shall have the value UNDEFINED. This parameter shall be omitted for a network-triggered Event Condition object.

15.4.1.2.7 Evaluation Interval

For a network-triggered Event Condition object, this parameter, of type integer, shall be omitted. If included it shall contain the value of the Evaluation Interval attribute of the Event Condition object.

15.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.4.2 Service Procedure

The VMD shall locate the Event Condition object identified by the specified Event Condition Name parameter and then issue a positive response containing the value of the specified attributes.

15.5 ReportEventConditionStatus Service

The ReportEventConditionStatus service is provided as the means whereby a client MMS-user may obtain the status of an Event Condition object from a VMD.

15.5.1 Structure

The structure of the component service primitives is shown in Table 71.

Table 71 – ReportEventConditionStatus Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Condition Name		M	M(=)			
Result(+)	(COMP)			S	S(=)	
Current State				M	M(=)	
Number Of Event Enrollments				M	M(=)	
Enabled				C	C(=)	
Time Of Last Transition To Active				C	C(=)	
Time Of Last Transition To Idle				C	C(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

15.5.1.1 Argument

This parameter shall convey the parameter of the ReportEventConditionStatus service request.

15.5.1.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the name of the Event Condition object for which the status report is requested.

15.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following parameters shall be returned in the response primitive.

15.5.1.2.1 Current State

This parameter, of type EC State, shall contain the current value of the Event Condition object's State attribute.

15.5.1.2.2 Number Of Event Enrollments

This parameter, of type integer, shall contain the current count of the number of entries in the Event Condition object's List Of Event Enrollment Reference attribute.

15.5.1.2.3 Enabled

This parameter, of type boolean, shall contain the value of the Enabled attribute of the Event Condition object, for a monitored Event Condition object. If the Event Condition Class attribute contains the value NETWORK-TRIGGERED, this parameter shall be omitted.

15.5.1.2.4 Time Of Last Transition To Active

For a monitored Event Condition object having a Time Of Last Transition To Active attribute with value not equal to UNDEFINED, this parameter, expressed as a date and time of day or a Time Sequence Identifier, shall contain the current value of the Time Of Last Transition To Active attribute. Otherwise, this parameter shall be omitted.

15.5.1.2.5 Time Of Last Transition To Idle

For a monitored Event Condition object having a Time Of Last Transition To Idle attribute with value not equal to UNDEFINED, this parameter, expressed as a date and time of day or a Time Sequence Identifier, shall contain the current value of the Time Of Last Transition To Idle attribute. Otherwise, this parameter shall be omitted.

15.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.5.2 Service Procedure

The VMD shall locate the Event Condition object identified by the specified Event Condition Name parameter and then issue a positive response, containing the current values for the specified attributes and the current count of entries in the List Of Event Enrollment Reference attribute.

15.6 AlterEventConditionMonitoring Service

This service is provided so that a client may request that the VMD alter any combination of a monitored Event Condition object's Enable, Priority, Alarm Summary Reports and Evaluation Interval attributes.

15.6.1 Structure

The structure of the component service primitives is shown in Table 72.

Table 72 – AlterEventConditionMonitoring Service

Table 72 (Cont.) – AlterEventConditionMonitoring Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Condition Name		M	M(=)			
Enabled		U	U(=)			
Priority		U	U(=)			
Alarm Summary Reports		U	U(=)			
Evaluation Interval		C	C(=)			CEI
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.6.1.1 Argument

This parameter shall convey the parameters of the AlterEventConditionMonitoring service request. At least one of Enable, Priority, Alarm Summary Reports or Evaluation Interval shall be present.

15.6.1.1.1 Event Condition Name

This parameter, of type Object Name, shall specify the name of the Event Condition object to be altered.

15.6.1.1.2 Enabled

This parameter, of type boolean and if included, shall specify the desired replacement value for the Event Condition object's Enabled attribute.

If the Enabled parameter is not specified, the Enabled attribute shall not be changed.

15.6.1.1.3 Priority

If included, this parameter, of type Priority, shall specify a replacement value for the Event Condition object's Priority attribute. A change in priority shall take effect immediately. and prior to execution of event transition processing.

If this parameter is omitted, the Priority attribute of the Event Condition object shall not be changed.

NOTE – Depending on the particular use of priority by an implementation, it is possible that assignment of increased priority may result in a period during which EventNotification requests primitives are not sequentially ordered with respect to time of occurrence.

15.6.1.1.4 Alarm Summary Reports

This parameter, of type boolean and if included, shall specify the desired replacement value for the Event Condition object's Alarm Summary Reports attribute.

If this parameter is not specified, the value of the attribute shall not be changed.

15.6.1.1.5 Evaluation Interval

This optional parameter, of type integer, shall, when included, convey a new proposed value for the Evaluation Interval attribute of the Event Condition object. This parameter shall not be included unless the CEI conformance block is supported by the VMD.

This parameter, when included, shall be provided as guidance to the VMD. Acceptance of this value shall guarantee only that the VMD shall make every effort to honour the specified value. A VMD which determines itself, through local means, to be unable to honour the service request, due to resource or other limitations, shall return a TIME-RESOLUTION error in response to the service request.

The actual decision to determine the value of the Event Condition object's State attribute may be based on the passage of time (such as a scan cycle), on knowledge about the variable referenced by the Monitored Variable parameter, or other local criteria. If the decision is to be time-based, then a non-zero value for the Evaluation Interval parameter shall specify the maximum acceptable time between determinations. A zero value shall indicate that any interval shall be acceptable.

15.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall not supply service specific parameters.

15.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.6.2 Service Procedure

The monitored Event Condition object specified by the Event Condition Name parameter shall be located, the requested changes shall be made and a response primitive specifying success shall be issued.

After the response is issued, and only if an included value for the Enabled parameter caused a change in the value of the Event Condition object's Enabled attribute, the value of the Event Condition object's State attribute shall be determined and the procedure for event transition processing (15.1.4.2) shall be executed for the indicated transition.

15.7 TriggerEvent Service

This service is provided so that a client may request that a VMD trigger an event associated with a network-triggered Event Condition object.

15.7.1 Structure

The structure of the component service primitives is shown in Table 73.

Table 73 – TriggerEvent Service

Table 73 (Cont.) – TriggerEvent Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Condition Name		M	M(=)			
Priority		U	U(=)			
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.7.1.1 Argument

This parameter shall convey the parameters of the TriggerEvent service request.

15.7.1.1.1 Event Condition Name

This parameter, of type Object Name, shall specify the name of the network-triggered Event Condition object which shall be triggered.

15.7.1.1.2 Priority

If included, this parameter, of type Priority, shall contain a replacement value for the Event Condition object's Priority attribute. A change in priority shall take effect immediately, and prior to execution of event transition processing.

If this parameter is omitted, the Priority of the Event Condition object shall not be changed.

NOTE – Depending on the particular use of priority by an implementation, it is possible that assignment of increased priority may result in a period during which EventNotification requests primitives are not sequentially ordered with respect to time of occurrence.

15.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall not supply service specific parameters.

15.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.7.2 Service Procedure

The network-triggered Event Condition object specified by the Event Condition Name parameter shall be located, the requested priority change shall be made (if applicable) and a response primitive specifying success shall be issued.

After the response is issued, the procedure for event transition processing (15.1.4.2) shall be executed for the specified Event Condition object.

15.8 DefineEventAction Service

The DefineEventAction service is provided as a mechanism whereby a client MMS-user may request the creation of an Event Action object at a VMD.

15.8.1 Structure

The structure of the component service primitives is shown in Table 74.

Table 74 – DefineEventAction Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Action Name	M	M(=)			
List Of Modifier	U	U(=)			
Confirmed Service Request (COMP)	M	M(=)			
Result(+) (COMP)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

15.8.1.1 Argument

This parameter shall convey the parameters of the DefineEventAction service request.

15.8.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value to be assigned to the Event Action Name attribute of the Event Action object created for the new event action. It shall uniquely identify the Event Action object.

15.8.1.1.2 List Of Modifier

This parameter shall contain the modifiers, if any, which apply for each execution of this event action, (see 5.4).

NOTE – This list does not apply to the execution of the DefineEventAction service.

15.8.1.1.3 Confirmed Service Request

This parameter shall contain a valid argument augmented by Companion Standard parameters, as specified by the Argument parameter of the request and indication primitives of a confirmed service. The responder role for this service shall have been included in the list of supported services negotiated by the VMD, through the use of the Initiate service, when the current Application Association was established.

15.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall not supply service specific parameters.

15.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.8.2 Service Procedure

If the Event Action object's definition is acceptable, an Event Action object shall be created and initialized as indicated below, and then a positive response shall be issued, indicating that the Event Action object has been created.

The initial values for the attributes of the newly created Event Action object are as follows:

Event Action Name - Initialized to the value of the Event Action Name parameter.

MMS Deletable - Initialized to the value true.

List Of Modifier - Initialized to the value of the List Of Modifier parameter, if present. Otherwise initialized to empty.

Confirmed Service Request - Initialized to the value of the Confirmed Service Request parameter.

List Of Event Enrollment Reference - Initialized to empty.

15.9 DeleteEventAction Service

The purpose for the DeleteEventAction service is to allow a client MMS-user to request that a VMD delete one or more Event Action objects.

15.9.1 Structure

The structure of the component service primitives is shown in Table 75.

Table 75 – DeleteEventAction Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Scope Of Delete		M	M(=)			
Event Action Names		C	C(=)			
Domain Name		C	C(=)			
Result(+)	(COMP)			S	S(=)	
Candidates Not Deleted				M	M(=)	
Result(−)				S	S(=)	
Error Type				M	M(=)	

15.9.1.1 Argument

This parameter shall convey the parameters of the DeleteEventAction service request.

15.9.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete to be attempted. This parameter shall contain one of the following values:

SPECIFIC - Shall designate the MMS defined Event Action objects identified by the value of the Event Action Names parameter as candidates for deletion.

AA-SPECIFIC - Shall designate all MMS defined Event Action objects whose scope is the current application association as candidates for deletion.

DOMAIN - Shall designate all MMS defined Event Action objects whose scope is the specified Domain as candidates for deletion.

VMD - Shall designate all MMS defined Event Action objects whose scope is VMD as candidates for deletion.

Only those candidate Event Action objects having a value of true for the MMS Deletable attribute and also having an empty List Of Event Enrollment Reference attribute shall be deleted.

15.9.1.1.2 Event Action Names

This parameter, containing one or more Event Action Names each of type Object Name and each identifying an Event Action object, shall be specified if the value of the Scope Of Delete parameter is equal SPECIFIC. Otherwise, it shall be omitted. It shall give the names of the specific candidate Event Action objects to be deleted.

15.9.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if the value of the Scope Of Delete parameter is equal to DOMAIN. Otherwise, it shall be omitted. It shall give the name of the Domain within which all MMS defined Event Action objects shall be designated as candidates for deletion.

15.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall contain the following parameter.

15.9.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain a count of the number of Event Action objects which satisfied the scope requirements of the Scope Of Delete parameter, but were not deleted because of a non-empty value of the List Of Event Enrollment Reference attribute, or a value of false for the MMS Deletable attribute. A deletion of some of those (Domain-specific, or AA-Specific) objects is still possible as a result of deletion of a Domain or loss of an application association.

15.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.9.2 Service Procedure

If the request is acceptable, for each Event Action object having an Event Action Name attribute of the specified scope, the responding VMD shall verify that the Event Action object's List Of Event Enrollment Reference attribute is empty. If not empty, or if the value of the MMS Deletable attribute is false, the VMD shall count the Event Action object as a Candidate Not Deleted, and shall not delete the Event Action object. Otherwise, the Event Action object shall be deleted and its name shall be made available for immediate redefinition.

Finally, a positive response shall be issued, indicating the count of candidate Event Action objects which were not deleted due to a non-empty value of the List of Event Enrollment Reference attribute, or a value of FALSE in the MMS Deletable attribute.

NOTE – If this count is not equal to zero, the requesting MMS-user (the client) may use the GetNameList service to determine the Event Action objects which were not deleted. Event Action objects can also become deleted as a result of the deletion of a Domain or loss of an application association. In these cases, the value of the Event Action Reference attribute of any Event Enrollment object which referred to a deleted Event Action object is set to UNDEFINED.

15.10 GetEventActionAttributes Service

The GetEventActionAttributes service shall provide a mechanism whereby a client MMS-user may obtain from a VMD the values of attributes of an Event Action object at the VMD.

15.10.1 Structure

The structure of the component service primitives is shown in Table 76.

Table 76 – GetEventActionAttributes Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Action Name	M	M(=)			
Result(+) (COMP)			S	S(=)	
MMS Deletable			M	M(=)	
List Of Modifier			M	M(=)	
Confirmed Service Request (COMP)			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

15.10.1.1 Argument

This parameter shall convey the parameters of the GetEventActionAttributes service request.

15.10.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the Event Action Name attribute of the Event Action object for which the attributes are requested.

15.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. The parameters of the Result(+) response primitive are described below.

15.10.1.2.1 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) this Event Action object may be deleted using the DeleteEventAction service.

15.10.1.2.2 List Of Modifier

This parameter shall contain the value of the Event Action object's List Of Modifier attribute. If no modifiers are specified for the event action, then an empty list shall be returned.

15.10.1.2.3 Confirmed Service Request

This parameter shall contain the value of the Event Action object's Confirmed Service Request attribute as augmented by Companion Standard parameters.

15.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.10.2 Service Procedure

The responding MMS-user shall locate the Event Action object identified by the value of the specified Event Action Name parameter and then issue a positive response containing the values of the specified attributes.

15.11 ReportEventActionStatus Service

The ReportEventActionStatus service may be used to obtain a count of the number of Event Enrollment objects which are currently specifying an Event Action object.

15.11.1 Structure

The structure of the component primitives is shown in Table 77.

Table 77 – ReportEventActionStatus Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Action Name	M	M(=)			
Result(+) (COMP)			S	S(=)	
Number Of Event Enrollments			M	M(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

15.11.1.1 Argument

This parameter shall convey the parameter of the ReportEventActionStatus service request.

15.11.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the Event Action Name attribute of the Event Action object for which status is desired.

15.11.1.2 Result(+)

This parameter shall indicate successful execution of the service procedure. The following parameter shall be included in the response primitive when success in indicated.

15.11.1.2.1 Number Of Event Enrollments

This parameter, of type integer, shall contain the current count of Event Enrollment references contained in the Event action object's List of Event Enrollment Reference attribute.

15.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.11.2 Service Procedure

The responding MMS-user shall locate the Event Action object identified by the value of the Event Action Name parameter and then issue a positive response containing the count of entries in the Event Action object's List of Event Enrollment Reference attribute.

15.12 DefineEventEnrollment Service

The DefineEventEnrollment service is provided as a means of allowing a client MMS-user to request that a VMD add the requesting client, or another, "third party" client, to the list of users for which the Procedure for Event Transition Processing (see 15.1.4.2) shall be executed as a result of a specified transition or set of transitions of an Event Condition object. The DefineEventEnrollment service shall cause the creation of a notification Event Enrollment object at a VMD. The DefineEventEnrollment service shall not be used to create a modifier Event Enrollment object.

15.12.1 Structure

The structure of the component service primitives is shown in Table 78.

Table 78 – DefineEventEnrollment Service

Table 78 (Cont.) – DefineEventEnrollment Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Enrollment Name	M	M(=)			
Event Condition Name	M	M(=)			
Event Condition Transitions	M	M(=)			
Alarm Acknowledgement Rule	M	M(=)			
Event Action Name	U	U(=)			
Client Application	U	U(=)			TPY
Acknowledgement Event Condition	U	U(=)			AKEC
Result(+) (COMP)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	
Object Not Defined			C	C(=)	

15.12.1.1 Argument

This parameter shall convey the parameters of the DefineEventEnrollment service request.

15.12.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the name to be used to reference the Event Enrollment object, and which shall become the value of the Event Enrollment Name attribute. This name must be unique among all Event Enrollment objects with identical scope. (VMD-specific, Domain-specific or AA-Specific.)

15.12.1.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the name of an existing Event Condition object for which an Event Enrollment shall be defined. The specified Event Condition object shall have been defined in the same abstract syntax as the abstract syntax of the application association over which the DefineEventEnrollment service request has been received.

15.12.1.1.3 Event Condition Transitions

The Event Condition Transitions parameter, of type Transitions, shall specify the set of transitions of the Event Condition for which invocation of the EventNotification service is requested. The allowed values for this parameter are specified in 15.1.3.1.

15.12.1.1.4 Alarm Acknowledgement Rule

The Alarm Acknowledgement Rule parameter, of type Alarm Ack Rule, shall specify the value of the Event Enrollment object's Alarm Acknowledgement Rule attribute. The semantics and allowed values for this parameter are given in 15.1.3.1.

15.12.1.1.5 Event Action Name

This optional parameter, of type Object Name, shall specify the value of the Event Action Name attribute of an existing Event Action object representing an action which shall be executed when the specified transition of the specified Event Condition occurs. The result of this execution shall be included in EventNotification requests initiated as a result of this Event Enrollment. The specified Event Action object shall have been defined in the same abstract syntax as the abstract syntax of the application association over which the DefineEventEnrollment service request was received. If omitted, the Event Enrollment object shall specify notification of event occurrence only.

15.12.1.1.6 Client Application

The semantics and value of this parameter, of type Application Reference, are as specified in 15.1.3.1. When included, it shall specify a client application, which may be the requester or a third-party application, to be enrolled on a permanent basis.

This parameter shall be specified when requesting creation of an Event Enrollment object with the value of the Duration attribute equal to PERMANENT, (permanent Event Enrollment object.) It shall be omitted when requesting creation of an Event Enrollment object with the value of the Duration attribute equal to CURRENT, (current Event Enrollment object.) This parameter shall not be specified unless the TPY conformance block is supported by the VMD.

15.12.1.1.7 Acknowledgement Event Condition

This optional parameter, of type Object Name, shall, when present, specify the value of the Event Condition Name attribute of a network-triggered Event Condition object which shall serve as an acknowledgement Event Condition object. This parameter shall only be present if the AKEC conformance block is supported by the VMD.

15.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

15.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.12.1.3.1 Object Not Defined

Object Not Defined, of type Object Name, shall be present when an access error pertains to the non-existence of the Event Condition object specified by the Event Condition Name parameter, or the Acknowledgement Event Condition parameter, or the Event Action object specified by the Event Action Name parameter. It shall provide the name of the object determined to be not defined at the VMD. This parameter shall not be present when the failure of this service is not due to the non-existence of an Event Condition object.

15.12.2 Service Procedure

The MMS server shall perform the following actions:

a) Verify that no Event Enrollment object exists with the same name as the Event Enrollment Name parameter;

b) Verify the existence of an Event Condition object with the same name as the Event Condition Name parameter;

c) Verify the existence of an Event Action object with the same name as the Event Action Name parameter, if present; and

d) Verify the existence of an Event Condition object with the same name as the Acknowledgement Event Condition parameter, if present.

If an Event Condition object or an Event Action object do not exist, then the responding MMS-user shall issue a Define Event Enrollment service response primitive with the Result(-) parameter with the Error Class of ACCESS, Error Code of OBJECT-NON-EXISTENT, and Object Not Defined parameter containing the value of the name which did not exist.

A notification Event Enrollment object for the requesting MMS-user (or the user specified by the Client Application parameter, if specified) shall be created, using the value of the Event Enrollment Name parameter for the value of the Event Enrollment Name attribute of the resultant Event Enrollment object. A reference to this Event Enrollment object shall then be placed in the List Of Event Enrollment Reference attribute for the specified Event Condition object and the Event Action object, if specified. If the Acknowledgement Event Condition parameter has been provided, the VMD shall create a reference to the resultant Event Enrollment object in the List Of Event Enrollment Reference attribute of the specified network-triggered Event Condition object. Finally a positive response shall be issued.

The attributes of the notification Event Enrollment object shall be initialized as follows.

a) Event Enrollment Name - Initialized to the value provided in the Event Enrollment Name parameter.

b) MMS Deletable - Initialized to the value true.

c) Enrollment Class - Initialized to the value NOTIFICATION.

d) Event Condition Reference - Initialized to reference the Event Condition object identified by the value of the Event Condition Name parameter.

e) Event Condition Transitions - Initialized to the value of the Event Condition Transitions parameter.

f) Application Association Local Tag - The abstract syntax component of this attribute is initialized to identify locally the abstract syntax in use on the application association over which the DefineEventEnrollment service request was received. The value of the application association local identifier component of this attribute shall be dependent on the presence or absence of the Client Application parameter. The following rules shall be in effect to determine the value of the application association local identifier component of this attribute:

   1) If the Client Application parameter has been omitted from the service request, thus specifying a current Event Enrollment object, this component shall be initialized to a value identifying locally the application association over which the DefineEventEnrollment request has been received.

   2) If the Client Application parameter has been included with the service request and the client to be enrolled is the same as the MMS-user initiating the DefineEventEnrollment service request, this component shall be initialized to a value identifying locally the application association over which the DefineEventEnrollment request has been received.

   3) If the Client Application parameter has been included, and the client to be enrolled is not the MMS-user submitting the DefineEventEnrollment service request, the client to be enrolled is defined as a "third party." If an application association exists with the third party client application to be enrolled which uses the same abstract syntax as the application association over which the DefineEventEnrollment service request was received, this component shall be initialized to a value identifying locally the existing application association.

   4) If the Client Application parameter has been included, and the client to be enrolled is not the MMS-user submitting the DefineEventEnrollment service request, the client to be enrolled is defined as a "third party." If an application association does not exist with the third party client application to be enrolled which uses the same abstract syntax as the application association over which the DefineEventEnrollment service request was received, this component shall be initialized to the value UNDEFINED.

   When the Client Application parameter is included, it shall specify a client application, which may be either the requester (rules 1 and 2) or a third party application (rules 3 and 4); see, for example, clause 15.12.1.1.6

g) Notification Lost - Initialized to the value false.

h) Event Action Reference - If the Event Action Name parameter was present in the service request, this attribute shall be initialized to reference the Event Action object identified by the value of the Event Action Name parameter. Otherwise, it shall be initialized to the value UNSPECIFIED.

i) Acknowledgement Event Condition Reference - Initialized to reference the network-triggered Event Condition object specified by the value of the Acknowledgement Event Condition parameter. Otherwise, it shall be initialized to the value UNSPECIFIED.

j) Duration - If the Client Application parameter was specified, this attribute shall be initialized to the value PERMANENT, indicating a permanent Event Enrollment object. Otherwise, it shall be initialized to the value CURRENT, indicating a current Event Enrollment object.

k) Client Application - Initialized to the value of the Client Application parameter, if specified. Otherwise this attribute shall be initialized to identify the client application for the current application association.

l) Alarm Acknowledgement Rule - This attribute is not applicable for an Event Enrollment object referencing a network-triggered Event Condition object. For an Event Enrollment object referencing a monitored Event condition object this attribute shall be initialized to equal the value of the Alarm Acknowledgement Rule parameter.

m) Time Active Acknowledged - This attribute is not applicable for an Event Enrollment object referencing a network-triggered Event Condition object. For an Event Enrollment object referencing a monitored Event Condition object this attribute shall be initialized to the value UNDEFINED.

n) Time Idle Acknowledged - This attribute is not applicable for an Event Enrollment object referencing a network-triggered Event Condition object. For an Event Enrollment object referencing a monitored Event Condition object this attribute shall be initialized to the value UNDEFINED.

o) State - The major state, included in the State attribute of the Event Enrollment object, shall be initialized to equal the value of the State attribute of the Event Condition object referenced by the Event Enrollment object. If the value of the Alarm Acknowledgement Rule parameter is not equal to NONE and the value of the major state is ACTIVE or IDLE, the minor state, included in the State attribute of the Event Enrollment object, shall be initialized to the value ACKED; otherwise there shall be no minor state.

15.13 DeleteEventEnrollment Service

The DeleteEventEnrollment service is provided as the means whereby a client MMS-user may request that a VMD delete one or more notification Event Enrollment objects. The DeleteEventEnrollment service shall not be used to delete modifier Event Enrollment objects.

15.13.1 Structure

The structure of the component service primitives is shown in Table 79.

Table 79 – DeleteEventEnrollment Service

Parameter Name	Req	Ind	Rsp	Ind	CBB
Argument (COMP)	M	M(=)			
Scope Of Delete	M	M(=)			
List of Event Enrollment Names	S	S(=)			
Event Condition Name	S	S(=)			
Event Action Name	S	S(=)			
Result(+) (COMP)			S	S(=)	
Candidates Not Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

15.13.1.1 Argument

This parameter shall convey the parameters of the DeleteEventEnrollment service request.

15.13.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete which is requested. It shall contain one of the following values.

EC - Shall designate as candidates for deletion Event Enrollment objects which:

a) are referenced in the List of Event Enrollment Reference attribute of the Event Condition object specified by the value of the Event Condition Name parameter, and

b) specify the requesting MMS-User in the Client Application attribute.

EA - Shall designate as candidates for deletion Event Enrollment objects which:

a) are referenced by the List of Event Enrollment Reference attribute of the Event Action object specified by the value of the Event Action Name parameter, and

b) specify the requesting MMS-User in the Client Application attribute.

SPECIFIC - Shall designate as candidates for deletion of the specific Event Enrollment objects listed in the List Of Event Enrollment Names parameter.

15.13.1.1.2 List of Event Enrollment Names

When the Scope Of Delete parameter specifies the value SPECIFIC, this parameter shall be included. Otherwise, it shall be omitted. When included, this parameter shall specify a list of one or more Event Enrollment Name attributes of Event Enrollment objects which shall be designated as candidates for deletion.

15.13.1.1.3 Event Condition Name

When the Scope Of Delete parameter specifies the value EC, this parameter, of type Object Name, shall be included and shall specify the value of the Event Condition Name attribute of the Event Condition object from which enrollments are to be identified as candidates for deletion. Otherwise, this parameter shall be omitted.

15.13.1.1.4 Event Action Name

When the Scope Of Delete parameter specifies the value EA, this parameter, of type Object Name, shall be included and shall specify the value of the Event Action Name attribute of the Event Action object from which enrollments are to be identified for deletion. Otherwise, this parameter shall be omitted.

15.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded and that the deletion process has been completed. A successful result shall not supply service specific parameters.

15.13.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain a count of the number of Event Enrollment objects which were included in the scope of Event Enrollment objects to be deleted, but which were not deleted due to a value of false in the MMS Deletable attribute. A deletion of those objects is still possible as a result of deletion of a Domain or loss of an application association.

15.13.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed and that none of the requested enrollments has been deleted. The Error Type parameter, which is defined in clause 17, shall provide the reason for failure.

15.13.2 Service Procedure

The specific procedure is dependent upon the value of the Scope Of Delete parameter. An Event Enrollment object having a value of MODIFIER for the Enrollment Class attribute shall not be deleted with the DeleteEventEnrollment service.

15.13.2.1 Scope Of Delete Equal To EC

When the value of the Scope Of Delete parameter is equal to EC, the VMD shall verify the existence of the specified Event Condition object. Then for each Event Enrollment object referenced in the Event Condition object's List Of Event Enrollment Reference attribute, and which:

a) specifies the requesting client MMS-user in the value of the Client Application attribute, and

b) has the value true in the MMS Deletable attribute

the VMD shall execute the procedure for Event Enrollment Deletion, (See 15.13.2.4.) Finally, a positive response shall be issued, indicating the count of candidate Event Enrollment objects which were not deleted due to the presence of the value false in the MMS Deletable attribute.

15.13.2.2 Scope Of Delete Equal To EA

When the value of the Scope Of Delete parameter is equal to EA, the VMD shall verify the existence of the specified Event Action object. Then for each Event Enrollment object referenced in the Event Action object's List Of Event Enrollment Reference attribute and which:

a) specifies the requesting client MMS-user in the value of the Client Application attribute, and

b) has the value true in the MMS Deletable attribute

the VMD shall execute the procedure for Event Enrollment Deletion, (See 15.13.2.4.) Finally, a positive response shall be issued, indicating the count of candidate Event Enrollment objects which were not deleted due to the presence of the value false in the MMS Deletable attribute.

15.13.2.3 Scope Of Delete Equal To SPECIFIC

When the value of the Scope Of Delete parameter is equal to SPECIFIC, the VMD shall verify the existence of each of the specified Event Enrollment objects. Then, if all specified Event Enrollment objects exist, for each of the specified Event Enrollment objects for which the value of the MMS Deletable attribute is equal to true, the VMD shall execute the procedure for Event Enrollment Deletion, (See 15.13.2.4.) Finally, a positive response shall be issued, indicating the count of candidate Event Enrollment objects which were not deleted due to the presence of the value FALSE in the MMS Deletable attribute.

15.13.2.4 Procedure for Event Enrollment Deletion

When an Event Enrollment object has been identified for deletion, the VMD shall remove references to the Event Enrollment object from the List Of Event Enrollment Reference attribute of the Event Condition object and (if applicable) the Event Action object referenced by the Event Enrollment object. Any invoked event notifications specifying the Event Enrollment object shall also be deleted, if possible.

If an Event Enrollment object having the value PERMANENT for the Duration attribute is deleted, and if the VMD was the Calling MMS-user on the application association being used to convey event notifications for the enrollment, then, as a local decision, the VMD may issue the Conclude service on that application association.

This procedure shall also be executed for deletion of Event Enrollment objects which result from deletion of a Domain or loss of an application association.

15.14 GetEventEnrollmentAttributes Service

The GetEventEnrollmentAttributes service is provided as a means whereby a client may request that a VMD return the values of descriptive attributes of an Event Enrollment object or a list of Event Enrollment objects which satisfy a specified set of criteria.

15.14.1 Structure

The structure of the component service primitives is shown in Table 80.

Table 80 – GetEventEnrollmentAttributes Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Scope Of Request		M	M(=)			
List of Event Enrollment Names		C	C(=)			
Client Application		C	C(=)			
Event Condition Name		C	C(=)			
Event Action Name		C	C(=)			
Continue After		U	U(=)			
Result(+)	(COMP)			S	S(=)	
List Of Event Enrollment				M	M(=)	
Event Enrollment Name				M	M(=)	
Event Condition Name				M	M(=)	
Event Action Name				C	C(=)	
Client Application				U	U(=)	
MMS Deletable				M	M(=)	
Enrollment Class				M	M(=)	
Duration				C	C(=)	
Invoke ID				C	C(=)	
Remaining Acceptable Delay				C	C(=)	
Acknowledgement Event Condition				C	C(=)	AKEC
Additional Detail				COMP	COMP	
More Follows				M	M(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

15.14.1.1 Argument

This parameter shall convey the parameters of the GetEventEnrollmentAttributes service request.

15.14.1.1.1 Scope Of Request

This parameter shall indicate the scope of Event Enrollment objects to be included in the request. It may specify one of four values:

SPECIFIC - Shall specify that values of the attributes for a list of Event Enrollment objects, specified in the value of the Event Enrollment Names parameter, is requested.

CLIENT - Shall specify that values of the attributes for a list of all Event Enrollment objects for which the specified client is enrolled (see Client Application, below) is requested.

EC - Shall specify that values of the attributes for all Event Enrollment objects referenced by the value of the List Of Event Enrollment Reference attribute of the specified Event Condition object (see Event Condition Name, below) is requested.

EA - Shall specify that values of the attributes of all Event Enrollment objects referenced by the value of the List Of Event Enrollment Reference attribute of the specified Event Action object (see Event Action Name, below) is requested.

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, then this parameter value shall be same as specified in the original request.

15.14.1.1.2 List of Event Enrollment Names

If this is a request to return values of the attributes of a specific Event Enrollment object or list of Event Enrollment objects, this parameter shall contain a list of values for the Event Enrollment Name attributes of the Event Enrollment objects for which the attributes are desired. If the value of the Scope Of Request parameter is other than SPECIFIC, this parameter shall not be included.

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, then this parameter value shall be same as specified in the original request.

15.14.1.1.3 Client Application

The meaning of this parameter, of type Application Reference, is dependent on the value of the Scope Of Request parameter.

If the Scope Of Request parameter specifies the value CLIENT, this parameter shall indicate the specific client application for which the request is made. If omitted, then the client for the current application association shall be implied.

If the Scope Of Request parameter specifies the values EC, EA or SPECIFIC, this parameter may be included to limit the result to only those Event Enrollment objects referenced by the List Of Event Enrollment Reference attribute of the specified Event Condition object or Event Action object (as applicable) and for which the specified client is enrolled. If omitted, all Event Enrollment objects of the applicable List Of Event Enrollment Reference attribute or attributes shall be included in the result without regard to their client.

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, then this parameter value shall be same as specified in the original request.

15.14.1.1.4 Event Condition Name

If the Scope Of Request parameter specifies the value EC, this parameter, of type Object Name, shall contain the value of the Event Condition Name attribute of the Event Condition object from which the list of Event Enrollment objects shall be determined.

Otherwise, this parameter shall be omitted.

15.14.1.1.5 Event Action Name

If the Scope Of Request parameter specifies the value EA, this parameter, of type Object Name, shall contain the value of the Event Action Name attribute of the Event Action object for which the list of Event Enrollment objects shall be determined. Otherwise this parameter shall be omitted.

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, then this parameter shall be the same as specified in the original request.

15.14.1.1.6 Continue After

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, then this parameter shall be specified. Otherwise, it shall be omitted. If specified, this parameter shall contain the value of the Event Enrollment Name parameter, of type Object Name, from the last entry in the List Of Event Enrollment parameter taken from the confirm primitive for which continuation is desired.

15.14.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameters.

15.14.1.2.1 List Of Event Enrollment

The List Of Event Enrollment parameter shall contain a list composed of listings of descriptive attributes of zero or more Event Enrollment objects satisfying the criteria of the request primitive. Each entry in the list shall contain the following parameters:

15.14.1.2.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment object's Event Enrollment Name attribute.

15.14.1.2.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the value of the Event Condition Name attribute from the Event Condition object referenced by the Event Condition Reference attribute of the indicated Event Enrollment object. If the referenced Event Condition object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association, this parameter shall have the value UNDEFINED.

15.14.1.2.1.3 Event Action Name

This parameter, of type Object Name, shall contain the value of the Event Action Name attribute from the Event Action object referenced by the Event Enrollment object's Event Action Reference attribute, unless it is UNSPECIFIED, in which case this parameter shall be omitted. If the Event Action object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association, this parameter shall have the value UNDEFINED, and shall be included. If the Enrollment Class attribute of the Event Enrollment object does not contain the value NOTIFICATION, this parameter shall be omitted.

15.14.1.2.1.4 Client Application

This parameter, of type Application Reference, shall contain the value of the Event Enrollment object's Client Application attribute, unless it specifies the requesting client, in which case this parameter shall be omitted. If the Enrollment Class attribute of the Event Enrollment object does not contain the value NOTIFICATION, this parameter shall be omitted.

15.14.1.2.1.5 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Event Enrollment object may be deleted using the DeleteEventEnrollment service.

15.14.1.2.1.6 Enrollment Class

This parameter, of type EE-Class, shall contain the value of the Event Enrollment object's Enrollment Class attribute.

15.14.1.2.1.7 Duration

This parameter, of type EE Duration, shall contain the value of the Event Enrollment object's Duration attribute. If the Enrollment Class attribute of the Event Enrollment object does not contain the value NOTIFICATION, this parameter shall be omitted.

15.14.1.2.1.8 Invoke ID

This parameter, of type integer and when present, shall contain the value of the Invoke ID attribute of the Event Enrollment object for a modifier enrollment.

If the Enrollment Class attribute does not contain the value MODIFIER, this parameter shall be omitted.

15.14.1.2.1.9 Remaining Acceptable Delay

This optional parameter, of type integer, shall convey the value of the Remaining Acceptable Delay attribute of the Event Enrollment object unless the value of this attribute is equal to FOREVER, in which case this parameter shall be omitted.

If the Enrollment Class attribute does not contain the value MODIFIER, this parameter shall be omitted.

15.14.1.2.1.10 Acknowledgement Event Condition

This optional parameter, of type Object Name, shall contain, when present, the value of the Acknowledgement Event Condition Reference attribute, unless the value is equal to UNSPECIFIED, in which case this parameter shall be omitted. If the AKEC conformance block is not supported by the VMD, this parameter shall be omitted.

15.14.1.2.1.11 Additional Detail

This parameter shall contain the current value of the Event Enrollment object's Additional Detail attribute, if present. Otherwise this parameter shall be omitted. The type of this parameter, if present, shall be defined in a Companion Standard.

If this parameter is present, then the abstract syntax of this instance of communication shall be the same as the abstract syntax of the Event Enrollment object's definition. If that abstract syntax is not available, then this parameter shall be omitted. If the Enrollment Class attribute of the Event Enrollment object does not contain the value NOTIFICATION, this parameter shall be omitted.

15.14.1.2.2 More Follows

This parameter, of type boolean, shall have the value true if this response does not contain attribute values from all of the requested Event Enrollment objects, and a continuation request shall be issued to obtain the additional event enrollment information. Otherwise, it shall be false.

15.14.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.14.2 Service Procedure

The service procedure varies depending upon the value of the Scope Of Request parameter. In each of the service procedures (specified below) the number of Event Enrollment objects which may be returned in the List Of Event Enrollment parameter may be limited by local restrictions. If the response does not contain all of the requested Event Enrollment objects the More Follows parameter shall be set to the value true in the response primitive. Otherwise, the value of the More Follows parameter shall be false.

15.14.2.1 Procedure for Scope of Request Equal to SPECIFIC

When the Scope of Request parameter has the value SPECIFIC, the VMD shall locate the specified Event Enrollment objects and return the list of the Event Enrollment objects' attributes.

15.14.2.2 Procedure for Scope Of Request Equal to CLIENT

When the Scope Of Request parameter has the value CLIENT, the responding MMS-User shall search all Event Condition objects for references to Event Enrollment objects specifying the indicated client in the value of the Client Application attribute and return a list containing the values of the required attributes of these Event Enrollment objects.

The search shall provide a well-defined ordering among Event Enrollment objects. The following order is specified:

a) Unless the Client Application parameter is included in the request, all AA-specific Event Condition objects, if any, shall be searched first, followed by VMD-specific Event Condition objects, followed by Domain-specific Event Condition objects. AA-specific Event Condition objects shall not be searched if the Client Application parameter is included in the request.

b) Within a given name scope, Event Condition objects shall be searched in increasing order, based on the collating sequence of the International Reference Version of ISO 646. For Domain-specific Event Condition objects, the Domain name shall be the major search key and the Domain-specific identifier shall be minor.

c) Within a given Event Condition object, all AA-specific Event Enrollment objects, if any, shall be searched first, followed by VMD-specific Event Enrollment objects, followed by Domain-specific Event Enrollment objects. Within each name scope, the search shall proceed in increasing order based on the collating sequence of the International Reference Version of ISO 646.

d) If the request contains the Continue After parameter, the search shall begin with the first Event Enrollment object (referenced in the List Of Event Enrollment Reference attribute of the specified Event Condition object) having an Event Enrollment Name attribute value logically greater than the Event Enrollment Name value provided in the Continue After parameter, based on the collating sequence.

15.14.2.3 Procedure for Scope Of Request Equal to EC

When the Scope Of Request parameter has the value EC, the VMD shall search the specified Event Condition object's List Of Event Enrollment Reference attribute and return a list containing only those Event Enrollment objects specifying the indicated Client Application (if specified) or all Event Enrollment objects (Client Application not specified).

Event Enrollment objects shall be returned in increasing order based on the collating sequence of the International Reference Version of ISO 646. If the request contains the Continue After parameter, the search shall begin with the first Event Enrollment object (referenced in the List Of Event Enrollment Reference attribute of the specified Event Condition object) having an Event Enrollment Name attribute value logically greater than the Event Enrollment Name value provided in the Continue After parameter, based on the collating sequence.

15.14.2.4 Procedure for Scope Of Request Equal to EA

When the Scope Of Request parameter has the value EA, the VMD shall search the specified Event Action object's List Of Event Enrollment Reference attribute and return a list containing only those Event Enrollment objects which, in the value of the Client Application attribute, specify the indicated client application (if specified) or all Event Enrollment objects (Client Application parameter not specified).

Event Enrollment objects shall be returned in increasing order of the Event Enrollment Name attribute, based on the collating sequence of the International Reference Version of ISO 646. If the request contains the Continue After parameter, the search shall begin with the first Event Enrollment object (referenced in the List Of Event Enrollment Reference attribute of the specified Event Condition object) having an Event Enrollment Name attribute value logically greater than the Event Enrollment Name value provided in the Continue After parameter, based on the collating sequence.

15.15 ReportEventEnrollmentStatus Service

The ReportEventEnrollmentStatus service is provided as the means whereby a client MMS-user may obtain the status of a single notification Event Enrollment object from a VMD.

15.15.1 Structure

The structure of the component service primitives is shown in Table 81.

Table 81 – ReportEventEnrollmentStatus Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Enrollment Name	M	M(=)			
Result(+) (COMP)			S	S(=)	
Event Condition Transitions			M	M(=)	
Notification Lost			M	M(=)	
Duration			M	M(=)	
Alarm Acknowledgement Rule			C	C(=)	
Current State			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

15.15.1.1 Argument

This parameter shall convey the parameter of the ReportEventEnrollmentStatus service request.

15.15.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment Name attribute of the Event Enrollment object for which the ReportEventEnrollmentStatus service is requested.

15.15.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following parameters are returned in the response primitive.

15.15.1.2.1 Event Condition Transitions

This parameter, of type Transitions, shall contain the current value of the Event Enrollment object's Event Condition Transitions attribute.

15.15.1.2.2 Notification Lost

This parameter, of type boolean, shall contain the current value of the Event Enrollment object's Notification Lost attribute.

15.15.1.2.3 Duration

This parameter, of type EE Duration, shall contain the value of the Event Enrollment object's Duration attribute.

15.15.1.2.4 Alarm Acknowledgement Rule

This parameter, of type Alarm Ack Rule, shall contain the current value of the Event Enrollment object's Alarm Acknowledgement Rule attribute.

15.15.1.2.5 Current State

This parameter, of type EE-State, shall report the value of the Reportable component of the State attribute of the Event Enrollment object.

15.15.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.15.2 Service Procedure

The responding MMS-user shall locate the specified notification Event Enrollment object. If the object is successfully located, the responding MMS-User shall issue a positive response, containing the current values for the specified Event Enrollment object's attributes. Otherwise, a Result(-) shall be issued.

15.16 AlterEventEnrollment Service

The AlterEventEnrollment service is provided as a means of allowing a client MMS-user to request that a VMD replace the value of the Event Condition Transitions attribute, or the value of the Alarm Acknowledgement Rule attribute, or both, of an existing notification Event Enrollment object. The AlterEventEnrollment service shall not be used to alter a modifier Event Enrollment object.

15.16.1 Structure

The structure of the component service primitives is shown in Table 82.

Table 82 – AlterEventEnrollment Service

Table 82 (Cont.) – AlterEventEnrollment Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Enrollment Name		M	M(=)			
Event Condition Transitions		U	U(=)			
Alarm Acknowledgement Rule		U	U(=)			
Result(+)	(COMP)			S	S(=)	
Current State				M	M(=)	
Transition Time				M	M(=)	
Result(-)				S	S(=)	
Error Type				M	M(=)	

15.16.1.1 Argument

This parameter shall convey the parameters of the AlterEventEnrollment service request.

15.16.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment Name attribute of the notification Event Enrollment object which is to be modified.

15.16.1.1.2 Event Condition Transitions

The Event Condition Transitions parameter, of type Transitions, shall specify the set of transitions of the Event Condition object for which invocation of the EventNotification service is requested. The allowed values for this parameter are specified in 15.1.3.1.

Either this parameter, or the Alarm Acknowledgement Rule parameter, or both, shall be specified.

15.16.1.1.3 Alarm Acknowledgement Rule

The Alarm Acknowledgement Rule parameter, of type Alarm Ack Rule, shall specify the value of the Event Enrollment object's Alarm Acknowledgement Rule attribute. The semantics and allowed values for this parameter are given in 15.1.3.1.

This parameter shall be omitted if the referenced Event Condition object is a network-triggered event condition.

Either this parameter, or the Event Condition Transitions parameter, or both, shall be specified.

15.16.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall supply the following additional parameters.

15.16.1.2.1 Current State

This parameter, of type EE State, shall contain the current value of the Reportable component of the State attribute of the Event Enrollment object following execution of the Service Procedure. If the Event Condition object from which the value of this attribute shall be determined has become unavailable, for example through deletion of a Domain or loss of an application association, the value of this parameter shall be set to UNDEFINED.

15.16.1.2.2 Transition Time

This parameter shall contain the time (date and time of day or Time Sequence Identifier) at which the transition to the current Event Enrollment object state was detected. If the execution of the AlterEventEnrollment service procedure resulted in alteration of the Reportable component of the Event Enrollment object's State attribute, this value shall reflect the time at which the value of the Reportable component of the State attribute was altered. If execution of the AlterEventEnrollment service procedure did not result in a change to the Reportable component of the State attribute, the value of the Transition Time parameter shall be equal to the value of the Time of Last Transition To Idle attribute of the referenced Event Condition or the value of the Time Of Last Transition To Active attribute of the referenced Event Condition object, whichever is later. If the referenced Event Condition object has become unavailable, through the loss of an application association or deletion of a Domain, the value of this parameter shall be UNDEFINED.

15.16.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.16.2 Service Procedure

The VMD shall locate the specified Event Enrollment object and replace the value of its Event Condition Transitions attribute, or the value of its Alarm Acknowledgement Rule attribute, or both, with the values specified in the Event Condition Transitions and Alarm Acknowledgement Rule parameters, respectively. Then the VMD shall issue a positive response. The value of the Current State parameter shall be UNDEFINED if the value of the Monitored Variable Reference attribute of the referenced Event Condition object is UNDEFINED.

NOTE – In the following discussion, the minor state NO-ACK-A should indicate that a required acknowledgement (given the new value for Alarm Acknowledgement Rule) for an EventNotification, specifying the ACTIVE state, has not been received. Likewise, the minor state NO-ACK-I should indicate that a required acknowledgement for an EventNotification, specifying the IDLE state, has not been received. The minor state ACKED should indicate that any acknowledgement required by the current value of the Alarm Acknowledgement Rule has been received. Minor states are described in 15.1.3.1.

If the specified value of the Alarm Acknowledgement Rule parameter is not equal to the value of this attribute prior to "replacement", then the minor state of the Event Enrollment object may be altered. Changes to the minor state of an Event Enrollment object shall be recorded in the State attribute of the Event Enrollment object, and reported in the Current State parameter. The minor state of an Event Enrollment object shall be determined as follows:

a) If the value of the Alarm Acknowledgement Rule parameter is equal to NONE, the value of the Event Enrollment object's State attribute shall be the same as the value of the State attribute of the referenced Event Condition object. (There is no minor state).

b) If the value of the Alarm Acknowledgement Rule parameter is equal to SIMPLE, then the major state of the Event Enrollment object shall be equal to the value of the State attribute of the referenced Event Condition object, and the minor state of the Event Enrollment object shall be determined as follows.

   1) If the major state is ACTIVE and the value of the Event Enrollment object's Time Active Acknowledged attribute is UNDEFINED, the minor state shall be NO-ACK-A.

   2) Otherwise, if the major state is ACTIVE, the minor state shall be ACKED.

   3) Otherwise, there shall be no minor state.

c) If the value of the Alarm Acknowledgement Rule attribute of the Event Enrollment object is equal to ACK-ACTIVE, then the major state of the Event Enrollment object shall be equal to the value of the State attribute of the referenced Event Condition object, and the minor state of the Event Enrollment object shall be determined as follows.

   1) If the major state is ACTIVE or IDLE, and the value of the Event Enrollment object's Time Active Acknowledged attribute is UNDEFINED, the minor state shall be NO-ACK-A.

   2) Otherwise, if the major state is ACTIVE or IDLE, the minor state shall be ACKED.

   3) Otherwise, there shall be no minor state.

d) If the Alarm Acknowledgement Rule is equal to ACK-ALL, then the major state of the Event Enrollment object is equal to the value of the State attribute of the referenced Event Condition object, and the minor state of the Event Enrollment object shall be determined as follows.

   1) If the major state is ACTIVE or IDLE, and the value of the Event Enrollment object's Time Active Acknowledged attribute is UNDEFINED, the minor state shall be NO-ACK-A.

   2) Otherwise, if the major state is IDLE, and the value of the Event Enrollment object's Time Idle Acknowledged attribute is UNDEFINED, the minor state shall be NO-ACK-I.

   3) Otherwise, the minor state shall be ACKED.

15.17 EventNotification Service

The EventNotification service is provided as a means whereby a VMD may notify an enrolled client of the occurrence of a state transition associated with an Event Condition object. The EventNotification service is an unconfirmed service.

15.17.1 Structure

The structure of the component service primitives is shown in Table 83.

Table 83 – EventNotification Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M(=)			
Event Enrollment Name	M	M(=)			
Event Condition Name	M	M(=)			
Severity	M	M(=)			
Current State	C	C(=)			
Transition Time	M	M(=)			
Notification Lost	M	M(=)			
Alarm Acknowledgement Rule	C	C(=)			
Action Result	C	C(=)			
Event Action Name	M	M(=)			
Success Or Failure	M	M(=)			
Confirmed Service Response (COMP)	S	S(=)			
Confirmed Service Error	S	S(=)			

15.17.1.1 Argument

This parameter shall convey the parameters of the EventNotification service.

15.17.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment Name attribute of the Event Enrollment object for which this notification is invoked.

15.17.1.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the value of the Event Condition Name attribute from the Event Condition object referenced by the Event Condition Reference attribute of the Event Enrollment object for which this EventNotification is invoked.

15.17.1.1.3 Severity

This parameter, of type integer, shall contain the value of the Severity attribute from the Event Condition object referenced by the Event Condition Reference attribute of the Event Enrollment object.

15.17.1.1.4 Current State

This parameter, of type EC State, shall contain the current value of the State attribute, following event transition processing, of the Event Condition object referenced by the Event Condition Reference attribute of the Event Enrollment object. If the EventNotification is being sent as a result of the transition ANY-TO-DELETED of the referenced Event Condition object, this parameter shall be omitted. This parameter shall also be omitted if the referenced Event Condition object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association.

15.17.1.1.5 Transition Time

This parameter shall contain the time (date and time of day or Time Sequence Identifier) at which the Event Condition object transition was detected.

15.17.1.1.6 Notification Lost

This parameter, of type boolean, shall contain the value of the Event Enrollment object's Notification Lost attribute at the time of the Event Condition object's transition. If true it shall indicate that one or more EventNotification requests specified for previous Event Condition object transitions associated with this Event Enrollment object have not been issued due to resource limitations in the issuing system. Following successful completion of an event notification, this parameter, if true, shall be set to the value false.

15.17.1.1.7 Alarm Acknowledgement Rule

This parameter, of type Alarm Ack Rule, shall be included for EventNotification service requests resulting from an Event Enrollment object referencing an Event Condition object which has a value for the Event Condition Class attribute of MONITORED. It shall contain the value of the Event Enrollment object's Alarm Acknowledgement Rule attribute.

This parameter shall be omitted for EventNotification service requests resulting from a network-triggered Event Enrollment object.

15.17.1.1.8 Action Result

This parameter shall be included for EventNotification service requests resulting from Event Enrollment objects which reference an Event Action object. Otherwise, it shall be omitted. If included, it shall contain the result of execution of the service request specified by the Confirmed Service Request attribute of the Event Action object. If the Event Action object has become unavailable, for example as a result of deletion of a Domain or loss of an application association, this parameter shall not be included. This parameter shall also not be included if the EventNotification is being sent as a result of the transition ANY-TO-DELETED of the referenced Event Condition object. The component parameters are specified as follows.

15.17.1.1.8.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the Event Action Name attribute from the Event Action object referenced by the Event Action Reference attribute in the Event Enrollment object.

15.17.1.1.8.2 Success Or Failure

This boolean parameter shall indicate whether the execution of the confirmed service specified in the Event Action object's Confirmed Service Request attribute succeeded (true) or failed (false).

15.17.1.1.8.3 Confirmed Service Response

This parameter shall contain the value of the Result(+) parameter augmented by Companion Standard parameters, resulting from the successful execution of the confirmed service specified in the Confirmed Service Request attribute of the Event Action object. It shall be present if the Success Or Failure parameter is true. Otherwise it shall be omitted.

15.17.1.1.8.4 Confirmed Service Error

This parameter shall contain the value of the Result(-) parameter resulting from the failure in execution of the confirmed service. It shall be present if the Success Or Failure parameter is false. Otherwise it shall be omitted.

15.17.2 Service Procedure

There is no response or confirm service primitive for the EventNotification service. However, depending on the presence and value of the Alarm Acknowledgement Rule parameter, the receiver of the service indication may be expected to initiate an AcknowledgeEventNotification service at some point following receipt of an EventNotification indication. The minimum requirements on the VMD for invocation of this service are specified in 15.1.4.

NOTE – The VMD, due to failure to receive a required acknowledgement or other local reason, may, at its discretion, repeat an EventNotification (in a new service invocation) for any Event Enrollment object which is awaiting a required acknowledgement. The criteria for deciding to repeat an EventNotification shall be determined as a local matter.

15.18 AcknowledgeEventNotification Service

The AcknowledgeEventNotification service is provided as the means whereby a client MMS-user may notify the VMD that its user (usually a human operator) has acknowledged an EventNotification received from the VMD.

This service shall be used to acknowledge EventNotification service indications containing the ACTIVE or IDLE value in the Current State parameter and containing a value for the Alarm Acknowledgement Rule parameter not equal to NONE.

This service may optionally be used to notify other clients enrolled for notifications of identical transitions of an Event Condition object, and receiving identical EventNotification service indications, through the use of an Acknowledgement Event Condition object. This object, which shall be a network-triggered Event Condition object, shall be triggered through the use of the optional Acknowledgement Event Condition parameter.

15.18.1 Structure

The structure of the component service primitives is shown in Table 84.

Table 84 – AcknowledgeEventNotification Service

Table 84 (Cont.) – AcknowledgeEventNotification Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Event Enrollment Name		M	M(=)			
Acknowledged State		M	M(=)			
Time Of Acknowledged Transition		M	M(=)			
Acknowledgement Event Condition		C	C(=)			AKEC
Result(+)	(COMP)			S	S(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.18.1.1 Argument

This parameter shall convey the parameters of the AcknowledgeEventNotification service request.

15.18.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall be equal in value to the Event Enrollment Name parameter of the EventNotification which is being acknowledged.

15.18.1.1.2 Acknowledged State

This parameter, of type EC State, shall be equal in value to the Current State parameter of the EventNotification which is being acknowledged.

15.18.1.1.3 Time Of Acknowledged Transition

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall be equal to the value of the Transition Time parameter of the EventNotification which is being acknowledged.

15.18.1.1.4 Acknowledgement Event Condition

This optional parameter shall be the name of the network-triggered Event Condition object which shall serve as the acknowledgement Event Condition. This parameter shall be included only if the AKEC conformance block is supported by the VMD.

15.18.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

15.18.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.18.2 Service Procedure

The VMD shall locate the Event Enrollment object identified by the value of the Event Enrollment Name parameter. The Procedure For Acknowledgement of Event Notifications (see 15.1.4.3) shall then be executed. If the Acknowledgement Event Condition parameter has been provided, the VMD shall locate the network-triggered Event Condition object specified by the value of this parameter and trigger it. The procedure For Event Transition Processing (15.1.4.2) shall be executed in response to the triggering of this object. Finally, the VMD shall issue a response primitive specifying success.

15.19 GetAlarmSummary Service

The GetAlarmSummary service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current status of monitored Event Condition objects and related attributes of their referenced notification Event Enrollment objects.

15.19.1 Structure

The structure of the component service primitives is shown in Table 85.

Table 85 – GetAlarmSummary Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Enrollments Only		M	M(=)			
Active Alarms Only		M	M(=)			
Acknowledgement Filter		M	M(=)			
Severity Filter		M	M(=)			
Most Severe		M	M(=)			
Least Severe		M	M(=)			
Continue After		U	U(=)			
Result(+)	(COMP)			S	S(=)	
List Of Alarm Summary				M	M(=)	
Event Condition Name				M	M(=)	
Severity				M	M(=)	
Current State				M	M(=)	
Unacknowledged State				M	M(=)	
Additional Detail				COMP	COMP	
Time Of Last Transition To Active				C	C(=)	
Time Of Last Transition To Idle				C	C(=)	
More Follows				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.19.1.1 Argument

This parameter shall convey the parameters of the GetAlarmSummary service request.

15.19.1.1.1 Enrollments Only

This parameter, of type boolean, shall provide a means of restricting the set of Event Condition objects which are to be summarized. When true, the summary shall include only those monitored Event Condition objects which contain (in the List of Event Enrollment Reference attribute) a reference to one or more notification Event Enrollment objects which specify (in the value of the Client Application attribute) the GetAlarmSummary service client.

15.19.1.1.2 Active Alarms Only

This parameter, of type boolean, shall provide a means of restricting the Event Condition objects which are to be summarized. When true, only those monitored Event Condition objects which have the value of the State attribute equal to ACTIVE shall be included. When false, monitored Event Condition objects shall be summarized without regard to the value of the State attribute.

15.19.1.1.3 Acknowledgement Filter

This parameter, of type integer, shall provide a means of restricting the Event Condition objects and Event Enrollment objects which are to be summarized. It may take any of three values:

NOTE – In the following discussions, an unacknowledged event enrollment is a notification Event Enrollment object in the minor state NO-ACK-I or in the minor state NO-ACK-A, and an acknowledged event enrollment is a notification Event Enrollment object in the minor state ACKED. A notification Event Enrollment object without a minor state is not included in either of these categories.

NOT-ACKED - Shall request that the GetAlarmSummary response shall contain only reports for those monitored Event Condition objects for which the List of Event Enrollment Reference attribute contains a reference to at least one unacknowledged event enrollment.

ACKED - Shall request that the GetAlarmSummary response contain only reports for those monitored Event Condition objects for which all references contained in the List of Event Enrollment Reference attribute are to acknowledged event enrollments.

ALL - Shall request that the GetAlarmSummary response contain monitored Event Condition objects without regard to the acknowledgement status of any referenced notification Event Enrollment objects.

15.19.1.1.4 Severity Filter

This parameter, containing two integers, is provided as a means of restricting the summary to monitored Event Condition objects having specific severity. Only those Event Condition objects for which the value of the Severity attribute is between Most Severe and Least Severe, inclusive, shall be summarized.

15.19.1.1.5 Continue After

This parameter shall be included if requesting continuation of a partially completed alarm summary, as indicated by a value of true in the More Follows parameter in the most recently received GetAlarmSummary confirm primitive. Otherwise, this parameter shall be omitted.

This parameter shall contain the Event Condition Name parameter, of type Object Name, shall be equal in value to the Event Condition Name parameter of the last Alarm Summary entry in the List Of Alarm Summary parameter of the GetAlarmSummary confirm service primitive for which continuation is requested.

15.19.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following parameters may also be included.

15.19.1.2.1 List Of Alarm Summary

This parameter shall contain a list of zero or more Alarm Summary parameters, each providing the summary of a single monitored Event Condition object satisfying the criteria of the GetAlarmSummary service indication.

15.19.1.2.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the value of the Event Condition Name attribute from the monitored Event Condition object.

15.19.1.2.1.2 Severity

This parameter, of type integer, shall contain the value of the monitored Event Condition object's Severity attribute.

15.19.1.2.1.3 Current State

This parameter, of type EC State, shall contain the current value of the monitored Event Condition object's State attribute.

15.19.1.2.1.4 Unacknowledged State

This parameter, of type integer, shall contain a value indicating the state or states of the monitored Event Condition object for which at least one referenced notification Event Enrollment object has not received a required acknowledgement. This parameter shall contain one of the following values:

NONE - shall indicate that no acknowledgements are outstanding.

ACTIVE - shall indicate that at least one acknowledgement of the most recent transition to the ACTIVE state is outstanding.

IDLE - shall indicate that at least one acknowledgement of the most recent transition to the IDLE state is outstanding.

BOTH - shall indicate that at least one acknowledgement of the most recent transition to the ACTIVE state and at least one acknowledgement of the most recent transition to the IDLE state are outstanding.

15.19.1.2.1.5 Additional Detail

Inclusion (or not) of this parameter shall be as specified in the Companion Standard regulating creation of the monitored Event Condition object. If the abstract syntax defined in ISO/IEC 9506-2, clause 19, was in effect for that creation, then this parameter shall be omitted.

15.19.1.2.1.6 Time Of Last Transition To Active

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the Time Of Last Transition To Active attribute of the monitored Event Condition object, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.19.1.2.1.7 Time Of Last Transition To Idle

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the Time Of Last Transition To Idle attribute of the monitored Event Condition object, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.19.1.2.2 More Follows

The More Follows parameter, of type boolean, shall be equal to true if this response does not contain all of the alarm summaries requested and a continuation request shall be issued to obtain the additional summaries. Otherwise, it shall be false.

15.19.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.19.2 Service Procedure

The responding MMS-user shall search all monitored Event Condition objects for which the value of the Event Condition Class attribute is MONITORED and for which the value of the Alarm Summary Reports attribute is true, for Event Condition objects which satisfy the specified filter criteria and shall return a positive response containing the List Of Alarm Summary parameter (constructed from monitored Event Condition objects found in this search) and the More Follows parameter.

If the request contains the Continue After parameter, the search shall begin with the first monitored Event Condition object logically greater than the monitored Event Condition object specified by the value of this parameter, according to the collating sequence specified below.

The search shall provide a well-defined ordering among monitored Event Condition objects. The following order is specified:

a) All AA-specific monitored Event Condition objects (of the requesting application association), if any, shall be searched first, followed by VMD-specific monitored Event Condition objects, followed by Domain-specific monitored Event Condition objects.

b) Within a given name scope, monitored Event Condition objects shall be searched in increasing order, based on the collating sequence of the International Reference Version of ISO 646. For Domain-specific monitored Event Condition objects, the Domain name shall be the major search key and the Domain-specific identifier shall be minor.

The number of monitored Event Condition objects which may be summarized in the List of Alarm Summary parameter may be limited by local restrictions. If the response does not contain all of the request alarm summaries, the More Follows parameter shall be set to the value true. Otherwise, the More Follows parameter shall be set to the value false.

15.20 GetAlarmEnrollmentSummary Service

The GetAlarmEnrollmentSummary service is provided as the means whereby a client MMS-user may request summary information from the VMD about the current alarm status of notification Event Enrollment objects and related attributes of their referenced monitored Event Condition objects.

15.20.1 Structure

The structure of the component service primitives is shown in Table 86.

Table 86 – GetAlarmEnrollmentSummary Service

Table 86 (Cont.) – GetAlarmEnrollmentSummary Service

Parameter Name		Req	Ind	Rsp	Cnf	CBB
Argument	(COMP)	M	M(=)			
Enrollments Only		M	M(=)			
Active Alarms Only		M	M(=)			
Acknowledgement Filter		M	M(=)			
Severity Filter		M	M(=)			
Most Severe		M	M(=)			
Least Severe		M	M(=)			
Continue After		U	U(=)			
Result(+)	(COMP)			S	S(=)	
List Of Alarm Enrollment Summary				M	M(=)	
Event Enrollment Name				M	M(=)	
Client Application				C	C(=)	
Severity				M	M(=)	
Current State				M	M(=)	
Additional Detail				COMP	COMP	
Notification Lost				M	M(=)	
Alarm Acknowledgement Rule				M	M(=)	
Enrollment State				C	C(=)	
Time Of Last Transition To Active				C	C(=)	
Time Active Acknowledged				C	C(=)	
Time Of Last Transition To Idle				C	C(=)	
Time Idle Acknowledged				C	C(=)	
More Follows				M	M(=)	
Result(–)				S	S(=)	
Error Type				M	M(=)	

15.20.1.1 Argument

This parameter shall convey the parameters of the GetAlarmEnrollmentSummary service request.

15.20.1.1.1 Enrollments Only

This parameter, of type boolean, shall provide a means of restricting the set of Event Enrollment objects which shall be summarized. When true, only those notification Event Enrollment objects for which the value of the Client Application attribute specifies the requesting MMS-user shall be summarized. When false, notification Event Enrollment objects shall be summarized without regard to the value of the Client Application attribute.

15.20.1.1.2 Active Alarms Only

This parameter, of type boolean, shall provide a means of restricting the notification Event Enrollment objects which shall be summarized. When true, only those notification Event Enrollment objects for which the value of the Major State component of the State attribute is equal to ACTIVE shall be included. When false, notification Event Enrollment objects shall be summarized without regard to the major state.

15.20.1.1.3 Acknowledgement Filter

This parameter, of type integer, shall provide a means of restricting the Event Enrollment objects which shall be summarized. It may take any of three values:

NOTE – In the following discussions, an unacknowledged event enrollment is a notification Event Enrollment object in the minor state NO-ACK-I or in the minor state NO-ACK-A, and an acknowledged event enrollment is a notification Event Enrollment object in the minor state ACKED. Event enrollments which do not have a minor state shall not be included in either of these categories.

NOT-ACKED - Shall request that the GetAlarmEnrollmentSummary response contain only reports for unacknowledged event enrollments.

ACKED - Shall request that the GetAlarmEnrollmentSummary response contain only reports for acknowledged event enrollments.

ALL - Shall request that the returned alarm enrollment summary contain reports of event enrollments for all notification Event Enrollment objects without regard to major or minor states.

15.20.1.1.4 Severity Filter

This parameter, containing two integers, is provided as a means of restricting the summary to notification Event Enrollment objects referencing Event Condition objects having specific severity. Only notification Event Enrollment objects referencing Event Condition objects for which the value of the Severity attribute is between Most Severe and Least Severe, inclusive, shall be summarized.

15.20.1.1.5 Continue After

This parameter, of type Object Name, shall be included if requesting continuation of a partially completed alarm enrollment summary, as indicated by a value of true in the More Follows parameter in the most recently received GetAlarmEnrollmentSummary confirm primitive. Otherwise, this parameter shall be omitted. When included, this parameter shall contain the value of the last Event Enrollment Name parameter included in the List Of Alarm Enrollment Summary response primitive preceding this request.

15.20.1.2 Result(+)

The Result(+) parameter shall indicate that the requested service has succeeded. When success is indicated the following parameters shall also be included.

15.20.1.2.1 List Of Alarm Enrollment Summary

This parameter shall contain a list of zero or more Alarm Enrollment Summary parameters, each providing the summary of a single notification Event Enrollment object satisfying the criteria of the GetAlarmEnrollmentSummary service indication.

15.20.1.2.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment Name attribute of the notification Event Enrollment object.

15.20.1.2.1.2 Client Application

This parameter, of type Application Reference, shall contain the value of the notification Event Enrollment object's Client Application attribute, unless it specifies the requesting client, in which case this parameter shall be omitted.

15.20.1.2.1.3 Severity

This parameter, of type integer, shall contain the value of the Severity attribute of the notification Event Enrollment object's referenced Event Condition object.

15.20.1.2.1.4 Current State

This parameter, of type EC State, shall contain the value of the State attribute of the notification Event Enrollment object's referenced Event Condition object.

15.20.1.2.1.5 Additional Detail

Inclusion (or not) of this parameter shall be as specified in the Companion Standard regulating creation of the notification Event Enrollment object's referenced Event Condition object. If the abstract syntax defined in this part of ISO/IEC 9506 was in effect for that creation, then this parameter shall be omitted.

15.20.1.2.1.6 Notification Lost

This boolean parameter shall contain the value of the notification Event Enrollment object's Notification Lost attribute.

15.20.1.2.1.7 Alarm Acknowledgement Rule

This parameter, of type Alarm Ack Rule, shall contain the value of the notification Event Enrollment object's Alarm Acknowledgement Rule attribute.

This parameter, of type EE-State, shall convey the value of the Reportable component of the State attribute of the notification Event Enrollment object. This parameter shall be omitted if the value of the Acknowledgement Filter parameter is not equal to ALL.

15.20.1.2.1.8 Time Of Last Transition To Active

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the Time Of Last Transition To Active attribute of the notification Event Enrollment object's referenced Event Condition object, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.20.1.2.1.9 Time Active Acknowledged

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the notification Event Enrollment object's Time Active Acknowledged attribute, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.20.1.2.1.10 Time Of Last Transition To Idle

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the Time Of Last Transition To Idle attribute of the notification Event Enrollment object's referenced Event Condition object, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.20.1.2.1.11 Time Idle Acknowledged

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the notification Event Enrollment object's Time Idle Acknowledged attribute, unless it has value UNDEFINED, in which case this parameter shall be omitted.

15.20.1.2.2 More Follows

The More Follows parameter, of type boolean, shall have a value of true if this response does not contain all of the alarm enrollment summaries requested and a continuation request shall be issued to obtain the additional summaries. Otherwise, the value shall be false.

15.20.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, shall provide the reason for failure.

15.20.2 Service Procedure

The responding MMS-user shall search the List Of Event Enrollment Reference attribute of all Event Condition objects for which the value of the Event Condition class attribute is MONITORED to obtain a list of notification Event Enrollment objects which satisfy the specified filter criteria and shall return a positive response containing the List Of Alarm Enrollment Summary parameter constructed from notification Event Enrollment objects found in this search and also containing the More Follows parameter.

If the request contains the Continue After parameter, the search shall proceed in accordance with the collating sequence specified below, commencing with the next Event Enrollment Name following the Event Enrollment Name contained in the Continue After parameter. If no such Event Enrollment Name exists, the search shall begin with the first Event Enrollment Name logically greater than the name specified by the Continue After parameter, based on the collating sequence.

The search shall provide a well-defined search ordering for monitored Event Condition objects and their referenced notification Event Enrollment objects. The following order is specified:

a) All AA-specific monitored Event Condition objects (of the requesting application association), if any, and their referenced notification Event Enrollment objects shall be searched first, followed by VMD-specific monitored Event Condition objects and their referenced notification Event Enrollment objects, followed by Domain-specific monitored Event Condition objects and their referenced notification Event Enrollment objects.

b) Within a given name scope, monitored Event Condition objects shall be searched in increasing order, based on the collating sequence of the International Reference Version of ISO 646. For Domain-specific monitored Event Condition objects, the Domain name shall be the major search key and the Domain-specific identifier shall be minor.

c) Within a given monitored Event Condition object the List Of Event Enrollment Reference attribute shall be searched in a manner identical with that specified for monitored Event Condition objects.

The number of notification Event Enrollment objects which may be summarized in the List of Alarm Enrollment Summary parameter may be limited by the server due to local considerations. If the response does not contain all of the requested summaries, the More Follows parameter shall be set to the value true. Otherwise, the More Follows parameter shall be set to the value false.

15.21 Attach To Event Condition Modifier

The Attach To Event Condition modifier is provided as the means whereby a client MMS-user may require that the VMD delay execution of a requested service's service procedure until a specified Event Condition object undergoes any one of a specified set of state transitions.

15.21.1 Structure

The structure of the Attach To Event Condition modifier parameter is shown in Table 87.

Table 87 – Attach To Event Condition Modifier

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Attach to Event Condition	S	S(=)			
Event Enrollment Name	M	M(=)			
Event Condition Name	M	M(=)			
Causing Transitions	M	M(=)			
Acceptable Delay	U	U(=)			

15.21.1.1 Attach To Event Condition

When a VMD implements the Attach To Event Condition service modifier the Attach To Event Condition parameter is provided as an alternative parameter of the List of Modifier parameter for each confirmed service request (see 5.4). The sub-parameters of the Attach To Event Condition modifier are specified as follows.

15.21.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment Name attribute which shall be used to identify the modifier Event Enrollment object. This name shall be unique among all other Event Enrollment Name attributes for Event Enrollment objects of identical scope at the VMD.

15.21.1.1.2 Event Condition Name

This parameter, of type Object Name, shall specify the value of the Event Condition Name attribute of the Event Condition object to which this service request or Event Action object is to be attached via a modifier enrollment.

15.21.1.1.3 Causing Transitions

This parameter, of type Transitions, shall specify the set of transitions of the Event Condition object which, individually, are to cause release of the service request or Event Action for continued execution of its service procedure. The allowed values for this parameter are as specified for the Event Condition Transitions attribute of the Event Enrollment object (15.1.3.1).

15.21.1.1.4 Acceptable Delay

This optional parameter, of type integer, shall indicate the duration of time in seconds for which the requesting MMS-User is willing to wait for the specified event condition to occur. If more than one Attach To Event Condition modifier has been specified, this parameter shall indicate for each modifier separately the duration of time which is acceptable after processing of that modifier has commenced. A zero value for this parameter is not acceptable and will result in a Result(-) response from the responding MMS-User. If no value is specified for this parameter, this is interpreted as meaning that any delay is acceptable, (i.e. "wait forever.")

15.21.2 Service Procedure

The responding MMS-user shall create a modifier Event Enrollment object having attributes as specified below and place a reference to this Event Enrollment object in the specified Event Condition object's List Of Event Enrollment Reference attribute. No further action shall be taken toward execution of the indicated service's service procedure until the specified Event Condition object undergoes one of the specified state transitions.

The attributes of the created Event Enrollment object shall be initialized as follows:

a) The Event Enrollment Name attribute shall be initialized to the value provided in the Event Enrollment Name parameter.

b) The Enrollment Class attribute shall be initialized to the value MODIFIER.

c) The Event Condition Reference attribute shall be initialized to reference the Event Condition object identified by the value of the Event Condition Name parameter.

d) The Event Condition Transitions attribute shall be initialized to the value of the Causing Transitions parameter.

e) The Invoke ID attribute shall be initialized to contain the invoke ID of the modified service and reference a Transaction object (see 7.2).

   In the case of an Event Action object specifying this modifier, this attribute shall have a value determined by local means. This value shall be unique with respect to values of Invoke IDs which may occur on the application association over which the Event Action object's definition was received.

f) If no value has been provided for the Acceptable Delay parameter, then the Remaining Acceptable Delay attribute shall be set to the value FOREVER. If a value has been provided, then the Remaining Acceptable Delay attribute shall be initialized to the value of the Acceptable Delay parameter and the timer function activated.

If the value of the Remaining Acceptable Delay attribute reaches zero, then the following actions shall be performed:

a) The timer shall be deactivated and the decrementation process shall cease;

b) The responding MMS-User shall return a Result(-) response for the modified service with the error parameter indicating expiration of acceptable delay;

c) The procedure for Event Enrollment Deletion (See 15.13.2.4) shall be followed.

If a specified state transition is detected, then the following actions shall be performed:

a) The service request shall be released for continued execution of its service procedure (See 7.2.11.2);

b) The Event Enrollment object and its reference in the specified Event Condition object's List Of Event Enrollment Reference attribute shall then be deleted (See 15.13.2.9.).

15.21.3 Procedure for Cancellation of Modified Service

This procedure amends the service procedure for the Cancel service when a service request attached to an Event Condition object is to be canceled. The procedure is as follows:

a) remove the reference to the Event Enrollment object from the List Of Event Enrollment Reference attribute of the Event Condition object referenced by the Event Enrollment object;

b) execute the service procedure specified for the Cancel service; then

c) delete the Event Enrollment object using the Procedure For Event Enrollment Deletion (See 15.13.2.4.).

Note that the cancellation of the modified service results in a negative response being issued for the service invocation.

15.22 Event Management State Diagrams

The state diagrams which regulate the use of the event management services are specified below. These state diagrams are in addition to state diagrams associated with the MMS-context. In the event that these diagrams are discovered to be in conflict with the preceding text, the text is considered to supersede the representations in the diagrams.

15.22.1 Event Condition State Diagrams

The state diagram for an event condition differs depending upon the class of the event condition.

15.22.1.1 Network-triggered Event Condition

The state diagram which regulates the network-triggered event condition is shown in Figure 15.

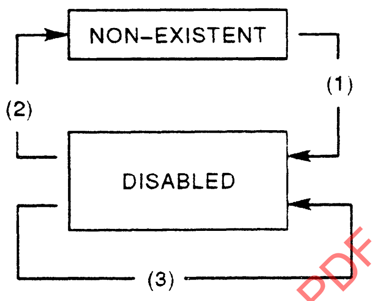


Figure 15 – Network-triggered Event Condition State Diagram

The arcs in this diagram are defined as follows.

1) Receive DefineEventCondition indication specifying a network-triggered event condition, or loss of application association if the Event Condition object is of AA-scope, or deletion of a Domain, if the Event Condition object is of Domain specific scope and dependent on the deleted Domain.

2) Receive DeleteEventCondition indication while List Of Event Enrollment attribute is empty, or loss of application association if the Event Condition object is of AA-specific scope and dependent on the lost application association, or deletion of a Domain if the Event Condition object is of Domain-specific scope and dependent on the deleted Domain.

3) Receipt of any of the following indications:

- DeleteEventCondition while List Of Event Enrollment is not empty.
- GetEventConditionAttributes;
- ReportEventConditionStatus;
- AlterEventConditionMonitoring;
- TriggerEvent;
- GetEventEnrollmentAttributes;
- DefineEventEnrollment;
- DeleteEventEnrollment;
- AcknowledgeEventNotification;
- GetAlarmSummary;
- GetAlarmEnrollmentSummary; or
- Service modified by Attach To Event Condition;

15.22.1.2 Monitored Event Condition

The state diagram which regulates a monitored event condition is defined by Figure 16.

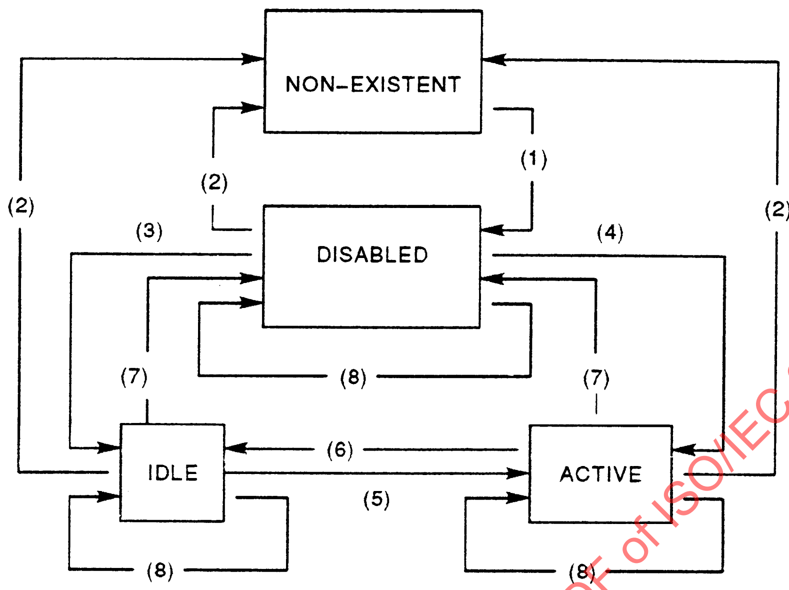


Figure 16 – Monitored Event Condition State Diagram

The arcs of Figure 16 are defined as follows.

1) Receive DefineEventCondition indication specifying a monitored event condition.

2) Receive DeleteEventCondition indication while List Of Event Enrollment Reference attribute is empty, or loss of application association if the Event Condition object is of AA-specific scope and dependent on the lost application association, or deletion of a Domain if the Event Condition object is of Domain-specific scope and dependent on the deleted Domain.

3) Receive AlterEventConditionMonitoring indication with Enable equal true while value of referenced monitored variable is false.

4) Receive AlterEventConditionMonitoring indication with Enable equal true while value of referenced monitored variable is true.

5) Value of referenced monitored variable changes from false to true.

6) Value of referenced monitored variable changes from true to false.

7) Receive AlterEventConditionMonitoring indication with Enable equal false.

8) Receipt of any of the following indications:

- DeleteEventCondition and List Of Event Enrollment Reference not empty.
- GetEventConditionAttributes;
- ReportEventConditionStatus;
- AlterEventConditionMonitoring with Enable omitted or equal to current value of the Enable attribute;
- TriggerEvent;
- GetEventEnrollmentAttributes;
- DefineEventEnrollment;
- DeleteEventEnrollment;

- GetAlarmSummary;
- GetAlarmEnrollmentSummary; or
- Service modified by Attach To Event Condition;

15.22.2 The Event Action State Diagram

The state diagram for an event action is shown in Figure 17.

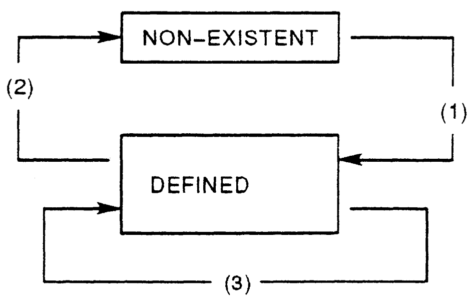


Figure 17 – Event Action State Diagram

The arcs in this diagram are defined as follows.

1) Receive DefineEventAction indication.

2) Receive DeleteEventAction indication while List Of Event Enrollment attribute is empty, or loss of application association if the Event Action object is of AA-specific scope and is dependent on the lost application association, or deletion of a Domain if the Event Action object is of Domain-specific scope and dependent on the deleted Domain.

3) Receipt of any of the following indications:

   - DeleteEventAction while List Of Event Enrollment attribute is not empty.
   - GetEventActionAttributes;
   - ReportEventActionStatus;
   - DefineEventEnrollment;
   - DeleteEventEnrollment; or
   - GetEventEnrollmentAttributes;

15.22.3 The Event Enrollment State Diagrams

The state diagrams of Figure 18 to 21 describe an event enrollment. The arcs shown in the various event enrollment state diagrams are defined as follows.

1) Receive AlterEventConditionMonitoring indication specifying Enable equal to true while variable referenced by Event Condition's Monitored Variable is true, or locally defined event is pending.

2) Receive AlterEventConditionMonitoring indication specifying Enable equal to true while variable referenced by Event Condition's referenced Monitored Variable is false, or locally defined event is not pending.

3) Variable referenced by Event Condition's referenced Monitored Variable becomes true or locally defined event is detected.

4) Variable referenced by Event Condition's Monitored Variable Reference attribute becomes false or condition causing locally defined event is cleared.

5) Receive AcknowledgeEventNotification for current state and time of transition.

6) Receive AcknowledgeEventNotification for ACTIVE state while in the IDLE state and time is equal to Event Condition's Time Of Last Transition To Active attribute.

7) Receive AlterEventConditionMonitoring indication specifying Enable equal to false.

Additional arcs, as specified in 15.11.1.2 are possible when an AlterEventEnrollment indication is processed which changes the value of the Alarm Acknowledgment Rule attribute. In this case the state diagram regulating the event enrollment is also changed.

Receipt of an indication for any of the following services does not change the state of an event enrollments.

- GetEventEnrollmentAttributes
- GetAlarmSummary
- GetAlarmEnrollmentSummary
- ReportEventEnrollmentStatus

Arcs indicating transitions that do not cause a state change are not shown in order to avoid clutter in the diagrams.

Receipt of a DefineEventEnrollment indication establishes a new event enrollment. This is modeled as a transition from the NON-EXISTENT state. The major state of the newly created event enrollment is equal to the current state of the referenced event condition. Its minor state is determined as specified above. Finally, receipt of an DeleteEventEnrollment indication is modeled as a transition from the current state to the NON-EXISTENT state. Arcs indicating transitions to or from the NON-EXISTENT state are not shown in order to avoid clutter in the diagrams.

15.22.3.1 Alarm Acknowledgment Rule equals NONE

Figure 18 contains the state diagram for an event enrollment specifying the Alarm Acknowledgment Rule equal to NONE.

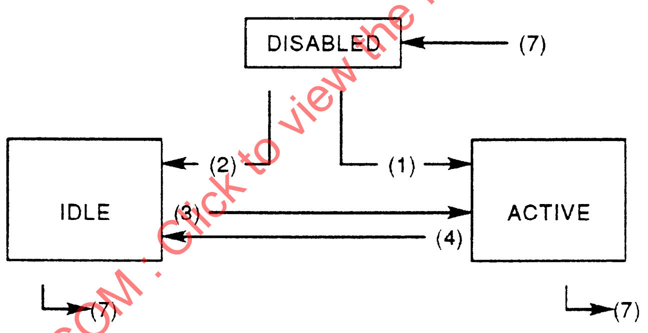


Figure 18 – State Diagram for Event Enrollment Alarm Acknowledgment Rule = NONE

15.22.3.2 Alarm Acknowledgment Rule equals SIMPLE

Figure 19 contains the state diagram for an event enrollment specifying the Alarm Acknowledgment Rule equal to SIMPLE.

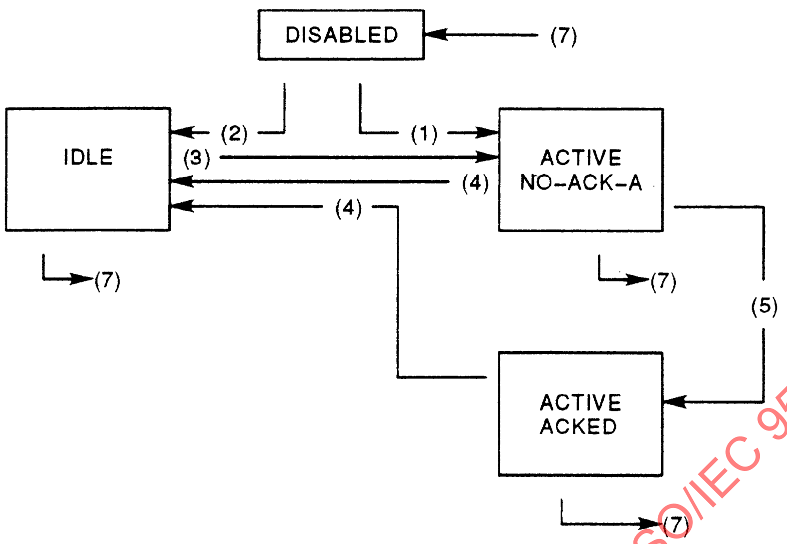


Figure 19 – State Diagram for Event Enrollment Alarm Acknowledgment Rule = SIMPLE

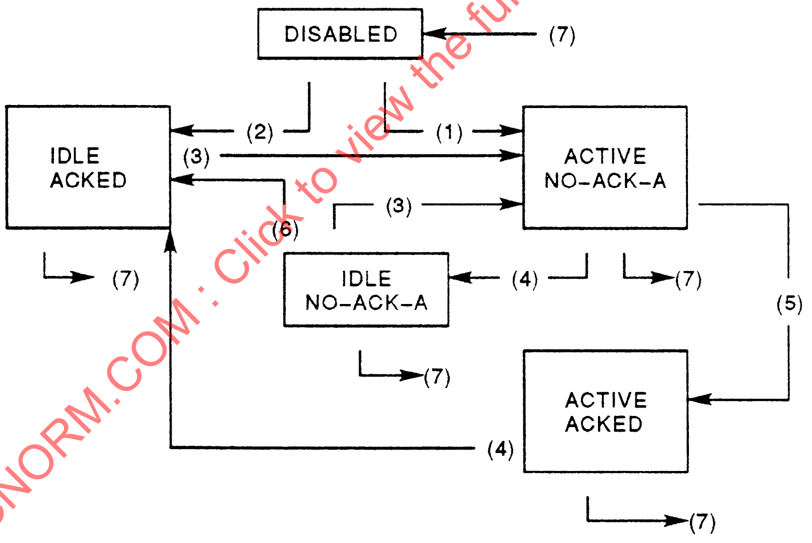


Figure 20 – State Diagram for Event Enrollment Alarm Acknowledgment Rule = ACK-ACTIVE

15.22.3.3 Alarm Acknowledgment Rule equals ACK-ACTIVE

Figure 20 contains the state diagram for an event enrollment specifying the Alarm Acknowledgment Rule equal to ACK-ACTIVE.

15.22.3.4 Alarm Acknowledgment Rule equals ACK-ALL

Figure 21 contains the state diagram for an event enrollment specifying the Alarm Acknowledgment Rule equal to ACK-ALL.

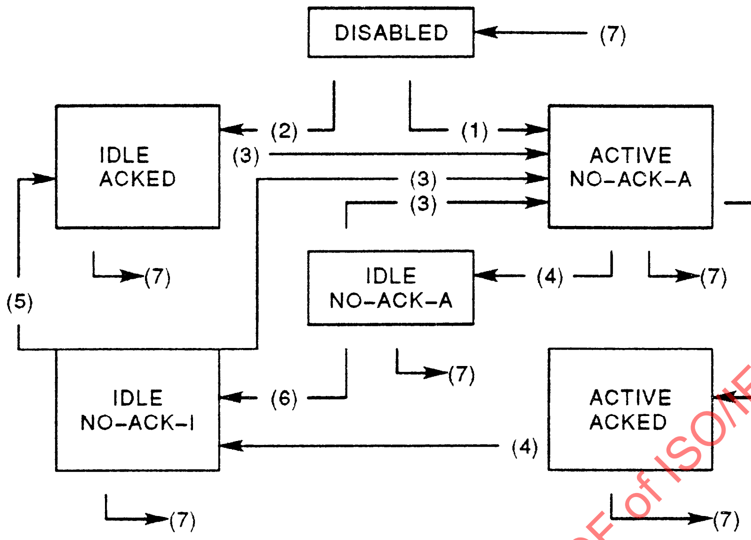


Figure 21 – State Diagram for Event Enrollment Alarm Acknowledgment Rule = ACK-ALL

15.23 Conformance Requirements Unique to Event Management

The Event Management Services define parameter and time-support requirements for the VMD. These requirements are described below.

15.23.1 Parameter Conformance Building Blocks

Parameter conformance to the MMS Event Management services is specified in terms of the requirements placed upon the VMD. These requirements are defined below

AKEC

The AKEC parameter conformance block shall establish the validity of the Acknowledgement Event Condition parameters of the DefineEventEnrollment and AcknowledgeEventNotification service arguments, and the GetEventEnrollmentAttributes service Result.

If AKEC is supported, these parameters are valid.

Otherwise, these parameters are invalid. A request or result specifying these parameters shall constitute a protocol error. A valid service request which, if honoured, would require a response specifying these parameters shall result in a service error specifying Error Class equal to ACCESS and Error Code equal to OBJECT-ACCESS-UNSUPPORTED.

CEI

The CEI parameter conformance block shall establish the validity of the Evaluation Interval parameter of the AlterEventConditionMonitoring service argument.

If CEI is supported, this parameter is valid.

Otherwise, this parameter is invalid. A request specifying this parameter shall constitute a protocol error.

15.23.2 Support for Time

The Protocol Implementation Conformance Statement for an implementation shall state the level of support for time (date and time of day, or Time Sequence Identifier.)

Support for the Time Sequence Identifier implies that an implementation will be capable of assigning a sequence number to all attributes which represent time, but that real time in terms of date and time of day is not supported. The relationship of the Time Sequence Identifier to date and time of day shall be for local determination.

Support for date and time of day implies that an implementation maintains a real time clock which is used to assign the values for all attributes which contain time (date and time of day.)

16 Journal Management Services

The purpose of the Journal Management Services is to provide a facility for the recording and retrieval of chronologically ordered information concerning events, variable contents of interest in conjunction with events, or both, and textual strings which may be used to provide, for example, annotating explanations or operator observations. The services contained in this clause are:

ReadJournal Service

WriteJournal Service

InitializeJournal Service

ReportJournalStatus Service

CreateJournal Service

DeleteJournal Service

16.1 The Journal Management Model

This clause defines the Journal object and the Journal Entry object, and provides services which operate on these objects. Journal objects may be predefined at an MMS server, or may be created through the use of the CreateJournal service. Journal objects shall have VMD-specific or AA-specific scope only. A Journal Entry object shall have a scope that is the same as the Journal object that it references via its Journal Reference attribute.

16.1.1 The Journal Object

NOTE – The MMS Journal object is considered to be capable of storing an indefinite number of Journal Entries. Although real journals deal with considerations such as storage media changeover and contents archival, these issues shall be local matters.

16.1.1.1 Attributes of a Journal Object

Object: Journal

```
Key Attribute: Journal Name
Attribute: MMS Deletable
Attribute: List Of Entry Reference
```

Journal Name

This attribute, of type Identifier, shall uniquely identify a particular Journal object.

MMS Deletable

This attribute shall indicate whether (true) or not (false) this object may be deleted through the use of the DeleteJournal service.

List Of Entry Reference

This attribute shall be a list of references to Journal Entry objects. Such Journal Entry objects contain the information that makes up the Journal. A Journal object shall contain zero or more entry references.

16.1.2 The Journal Entry Object

A Journal Entry object is a time-stamped record of information within a Journal object.

16.1.2.1 Attributes of a Journal Entry Object

Object: Journal Entry

```
Key Attribute: Journal Reference
Key Attribute: Entry Identifier
Attribute: Application Process Identification
Attribute: Time Stamp
Attribute: Order Of Receipt
Attribute: Additional Detail
Attribute: Information Type (ANNOTATION, EVENT-DATA, DATA)
Constraint: Information Type = ANNOTATION
    Attribute: Textual Comment
Constraint: Information Type = EVENT-DATA
    Attribute: Event Transition Record
    Attribute: List of Journal Variables
Constraint: Information Type = DATA
    Attribute: List of Journal Variables
```

Journal Reference

This attribute shall uniquely identify the Journal object with which the Journal Entry object is associated. This attribute shall be invariant once assigned.

Entry Identifier

This attribute shall be of type octet string and shall be one to eight octets in length. The value of this attribute shall be assigned by the server such that no two Journal Entry objects whose Journal Reference attributes are equal have the same value for this attribute. This attribute shall be invariant once assigned. The primary use of this value is to resolve the identity of a particular Journal Entry object during access where multiple Journal Entry objects have an identical time-stamp.

Application Process Identification

This attribute, of type Application Reference, shall identify the Application Process that caused the server to create the Journal Entry object.

Time Stamp

This attribute, of type TimeOfDay and whose value shall be specified by the Application Process identified in the Application Process Identification attribute, shall contain a time of day.

NOTE 1 – This attribute is intended to be used by a client to record the time of occurrence associated with the Textual Comment, Event Transition Record and List of Journal Variables, or List of Journal Variables attributes for the Journal Entry object.

Order Of Receipt

This attribute represents the order of creation of the Journal Entry object among all other Journal Entry objects with the same Time Stamp attribute value and same Journal Reference attribute value. Journal Entry objects are created when they are received and made a part of a Journal.

The intent of this attribute is to provide an ordering among Journal Entry objects within a journal that have the same time stamp. The server shall assign the value of the Order Of Receipt attribute so as to be monotonically increasing by order of receipt of the information in the entry for those entries with the same Time Stamp and Journal Reference.

Additional Detail

This attribute contains zero or more attributes whose meaning and syntax shall be defined by Companion Standards.

Information Type

This attribute shall indicate the type of information contained in the Journal Entry object. If the value of this attribute is ANNOTATION, the information in the Journal Entry object shall represent a textual comment (which may be used to document or comment on some condition or set of conditions). If the value of this attribute is EVENT-DATA, the information in the Journal Entry object shall represent a record of the occurrence of an event and zero or more variable values. If the value of this attribute is DATA, the information in the Journal Entry object shall represent zero or more variable values.

Textual Comment

This attribute, which shall exist only if the Information Type attribute value is ANNOTATION, shall contain a textual comment. This attribute shall be a character string whose length is between zero and two hundred fifty five (255) characters, inclusive.

NOTE 2 – This attribute is intended to be used by a client to document or comment on some condition or set of conditions.

Event Transition Record

This attribute, which shall exist only if the Information Type attribute value is EVENT-DATA, shall contain an Event Condition Name and an Event Condition State (see 15.1.1.1).

NOTE 3 – This attribute is intended to be used by a client to record the occurrence of an event by specification of the Event Condition Name and the resulting State attribute of this Event Condition.

List of Journal Variables

This attribute, which shall exist only if the Information Type attribute value is EVENT-DATA or DATA, shall contain zero or more journal variables. Each journal variable shall contain a variable tag, which shall be a character string whose length shall be not more than 32 characters, and a data value as specified by 12.4.2.

NOTE 4 – If the Information Type attribute value is EVENT-DATA, this attribute is intended to be used by a client to record the values of zero or more variables at the time of the occurrence of the event specified in the Event Transition Record.

16.1.3 Operations on Journal and Journal Entry Objects

Services which operate on Journal and Journal Entry objects are:

a) ReadJournal: retrieves one or more Journal Entry objects referenced by a specified Journal object;

b) WriteJournal: causes one or more Journal Entry objects to be added to a journal by causing each such Journal Entry object to reference the Journal object, and the Journal object to reference each such Journal Entry object;

c) InitializeJournal: causes zero or more Journal Entry objects that are referenced by a Journal object to be deleted, thereby clearing all or part of that journal;

d) ReportJournalStatus: returns the number of Journal Entry objects referenced by a specified Journal object and whether the Journal object specified may be deleted;